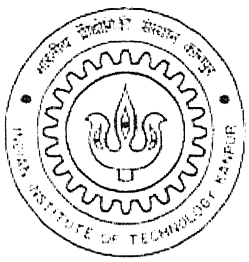
Dimensional Synthesis of Kinematically Redundant Spatial Manipulators

A Thesis Submitted
in Partial Fulfilment of the Requirements
for the degree of

Master of Technology

by

Surya Mani Tripathi



to the

DEPARTMENT OF MECHANICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KANPUR

June, 2004

Dedicated to

My Late Mother and Sister

CERTIFICATE

It is certified that the work contained in this thesis entitled "**Dimensional Synthesis of Kinematically Redundant Spatial Manipulators**," by **Mr. Surya Mani Tripathi (Roll No. Y210549)**, has been carried out under my supervision and this work has not been submitted elsewhere for a degree.

Bhaskar Dasgupta

Dr. Bhaskar Dasgupta,
Assistant Professor,
Department of Mechanical Engineering,
Indian Institute of Technology, Kanpur
Kanpur, 208016.

Acknowledgement

With immense pleasure I express my sincere gratitude, regards and thanks to my thesis supervisor Dr. Bhaskar Dasgupta for his excellent guidance, invaluable suggestions, continuous support and encouragement during the tenure of this thesis work. It was great pleasure to work under him, as a lot of care with personal touch was rendered.

It was of great pleasure to be associated with Center for Robotics and would like to thank all the research engineers, staff members and my colleagues Hari, Madan, Divya, Pankaj, Prasad, Sandeep, Shrikant, Srinivas, Shahzad, Rakesh, Vivek, Siddharth, Parthajit and Omkar for encouraging me in all stages of my work and making the Center for Robotics a great place to work in.

I wish to acknowledge all my friends, batchmates and faculties, who were a part of the process of intellectual enrichment and personal growth in many aspects of life.

Surya Mani Tripathi

IIT Kanpur

Abstract

Synthesis problem of mechanisms is dealt with extensively in literature, but presence of redundancy in system makes it more interesting. In this work, the problem of point-to-point motion of spatial kinematic redundant manipulator working in environment with polyhedral obstacles is considered. The manipulators considered for analysis are assumed to have revolute/prismatic or combination of both types of joints.

The problem is formulated as a constrained optimization problem minimizing positional error and simultaneously avoiding the obstacles. The obstacle avoidance is carried out using the proximity queries which in turn utilizes the fast proximity queries with swept sphere volumes for preparing bounding volume hierarchy. The techniques used to tackle the optimization problem encompass both the classical Augmented Lagrangian Method and evolutionary technique of Genetic Algorithm.

The proficiency of the formulation is shown by means of several results on different spatial kinematic redundant manipulators for different environments.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Redundant Manipulators

Redundant manipulators are a class of manipulators which have more degrees of freedom (DOF) than the required minimum to perform a given task in a specified workspace. Different kinds of redundancy that may be present in a redundant manipulator are

1. **Kinematic**
2. **Force**
3. **Actuation**
4. **Sensor**

In the present work, kinematically redundant manipulators, in which links are connected serially, have been considered. These manipulators have more number of links (hence, more DOF) than the required minimum to perform a task in the given workspace. A good example of a redundant manipulator is the all-too familiar human hand.

The implementation of redundant manipulators is justified due to improved capabilities shown by them. Due to the use of property of redundancy we achieve enhancement in different properties of manipulators like reachability, dexterity, workspace volume and payload capacity. The redundancy present may also be utilised for obstacle avoidance and fault tolerance.

1.2 Synthesis

The synthesis of the redundant manipulator is the procedure of finding out dynamic and kinematic parameters of a manipulator which satisfy certain requirements. Dynamic parameters may be torque, acceleration and velocity at the joints, whereas kinematic parameters deal with type, number and geometry of the manipulator. In the present work geometric synthesis of spatial redundant manipulators has been considered. This involves finding out optimal geometric parameters such as link lengths, shapes, configurations and minimum number of links to obtain the optimal results for manipulator kinematics.

Synthesis of redundant manipulators is more difficult than that of a non-redundant maniplator. In this case, the increasing number of joint variables are the major cause of involved complexity. As the number of links increases, the inverse kinematics becomes difficult from the viewpoint of number of solutions and computational complexity.

A lot of effort has been devoted in the quest for developing efficient procedures to solve the inverse kinematics problem of redundant manipulators. The presence of redundancy gives infinite solutions, some of which are good and others are either ill-conditioned or not suitable for the purpose. Therefore, choosing a suitable solution out of these infinite solutions itself is a non-trivial problem. These infinite solutions arise becuase of the mapping, which takes place from the joint space coordinates to the task space coordinates, is not a one-to-one mapping.

Turning our attention to computational complexity, it has been observed that even to follow a given trajectory for the end-effector, path-planning is required for the joints and time taken in this process varies exponentially as the number of degrees of freedom, as shown by Hopcroft *et al.* [1] and Sedgewick [2].

Hence synthesis of a redundant manipulator has to incorporate the various parameters as constraints. Among these, the first one may be the ability of the manipulator to reach the specified task space locations (TSL's). This is taken as the primary priority for the manipulator by Lenarčič [3] and Nakamura *et al.* [4]. The secondary priority for the synthesis can be anything like obstacle avoidance, singularity avoidance, dexterity of the manipulator and several other

requirements which affect the kinematic design of manipulator directly or indirectly to some extent.

There may be cases when, some of the above mentioned requirements may be conflicting with other requirements. For example, in the task-priority method, the desired increment in the secondary task-space is projected in the null space of the Jacobian matrix associated with the primary task. This projection may distort the projected vector and, therefore, secondary task may not be solved efficiently. Thus the efforts to improve one is hampering other, so a trade-off is made between two conflicting priorities. Since, in most of the cases, the objective of any synthesis is to fulfil the primary priority at any cost, so in this respect the secondary priority can be satisfied to some acceptable limits. The problem of distortion of the projected vector can be minimized in two different ways. First is to optimize the weight of the pseudo-inverse matrix, but the drawback of this is that it simultaneously influences the execution of the primary task. The second way to do this is to directly optimize the null space matrix by minimizing the difference between the projected and the desired vector of the secondary task increment. The advantage of the second method is that it does not interfere with the execution of the primary task (refer Lenarčič [3]).

In the light of the above, the synthesis problem at hand is tackled by a direct kinematics approach based on optimization techniques. The advantages and shortcomings of this particular idea has been emphasized by the results presented in this work. Some of the landmark works which pave the way ahead for the problem of synthesis of redundant manipulators in some aspects, have been reviewed briefly in the following section.

1.3 State of the Art

The problem of synthesis is historically significant. The roots of this may be traced back to the date when synthesis of piston mechanism for steam engine was performed. This problem initially developed as an art with time, but today it is a vast field of interest in which geometric and dynamic aspects for a vast range of mechanisms is considered. Different graphical, vector, tensor, neural network and optimization based techniques have been proposed by researchers

for differtent kinds of mechanisms. In spite of all this, synthesis of complex system like redundant manipulators remains a challenging task even today.

Numerous researchers have tackled the synthesis problem by studying different aspects of manipulators and have proposed many concepts and algorithms. A general study of kinematically redundant manipulator structure was done by Waldron *et al.* [5]. Klein [6] anlysed the problem of using redundancy in design of robot systems. Nakamura *et al.* [4] proposed the concept of task priority in relation to inverse kinematics and utilized it in a manner that subtasks of lower priority can be performed utilizing the redundancy of manipulators. These subtasks may be obstacle avoidance, manipulability, dexterity etc. Yoshikawa [7, 8] proposed the concept of dynamic manipulability of robot mechanisms and analysed it for manipulators. This concept is being utilized by several researchers as one of the priorities to be attained for synthesis. Well beyond this, Maciejewski *et al.* [9] considered the problem of the inverse kinematics of redundant robots in an environment with moving obstacles. The problem is posed as an optimization problem, with trajectory following as the primary goal and obstacle avoidance as the secondary one. Following the track, Baillieul [10] performed the design of kinematically redundant manipulators based on measure of *manipulability index*. Meanwhile, an important concept of reconfigurable manipulator system was floated by Lee *et al.* [11] wherein a manipulator can adjust its mechanical structure to suit the kinematic characteristics of a given task. Later, an optimal synthesis procedure for reconfigurable manipulators was given by Paredis *et al.* [12], in which links of various length, were reconfigured to achieve various task specifications such as reachability, joint angle limits and obstacle avoidance etc. and Paredis *et al.* [13], elaborated how the manipulator modules can be reconfigured to perform a specific task and synthesized for a fault tolerant system. They concluded that for a planar manipulator, $2k$ redundant degree of freedoms (DOFs) are necessary and sufficient for k-th order fault tolerance. Maciejewski *et al.* [14, 15] discussed how the redundancy can be utilized for anticipating fault tolerance (in locked and free-swinging joints) in redundant manipulators.

Many authors used gradient based techniques to solve the synthesis problem as an optimization problem but finding the gradients of different functions involved, raised new questions.

Singh *et al.* [16] emphasized the difficulties in finding out the exact gradient for optimization and suggested a remedy to solve the synthesis problem of non-redundant manipulators using gradient based optimization methods. At the same time Mariappan *et al.* [17] used the optimal manipulability and isotropy values from global velocity ellipsoid to solve the synthesis problem using the generalized exact gradient based method for non-redundant manipulators. A recent work by Jimenez *et al.* [18] proposed a simple and general method for kinematic synthesis of spatial mechanism. The formulation is fully based on a set of *Cartesian Coordinates.* Using these coordinates the system to be synthesized is fully described by means of a set of *geometric constraints* and the design requirements are introduced by a set of *functional constraints.* Finally an objective function is defined and minimized to obtain the value of design parameters. Contemporary to the above work, some Neural-Network based quick techniques had been proposed to solve the inverse kinematic problem. This was to make the obstacle avoidance real-time in case of redundant manipulators. Han *et al.* [19], utilized *Tank-Hopfield networks* and *J functions* in the framework of resolved motion rate control for real-time implementation.

Wenger *et al.* [20] proposed some guidelines for the designers of new manipulator so that in the process of synthesis, there can be the possibility to adjust the kinematic parameters as function of three unusual but important kinematic properties like *cuspidality*[1], *genericity*[2] and *solvability*[3]. This work is expected to be of use for alternative manipulator design. Parker *et al.* [21] introduced the use of Genetic Algorithms for solving the inverse kinematics problem of redundant robots to position the end-effector of a robot at a target location while minimizing the largest joint displacement from the initial postion. Nearchou *et al.* [22], used evolutionary techniques for the problem of point-to-point motion of redundant manipulators working in environments with obstacles. They used the technique of binary-coded Genetic Algorithms to optimize the problem imposing a penalty on the objective function to avoid obstacle. In this work obstacle avoidance had been acompolished by a scheme based on the concept of convex hulls. Authors claimed very good efficiency of the method and have shown convincing results

[1]non-singular posture changing ability
[2]stability of the kinematic properties with repect to small variations in the design parameters
[3]the inverse kinematic problem can be solved with quadratics

for planar manipulators and expected the algorithm to work for spatial case too.

All these works give good background for solving the inverse kinematic problem of synthesis based on different techniques and many proposals for optimal synthesis, obstacle avoidance, manipulability measurement etc. But, more or less, these concepts are specific to some problem and have certain limitations of their own.

In this work, a different methodology is adopted for a more general approach for the synthesis of redundant manipulators, which utilizes optimization techniques of Augmented Lagrangian and real-coded Genetic Algorithms. Obstacle avoidance is based on a scheme of fast proximity queries proposed by Gottschalk et al. [23, 24] which is very quick and general.

1.4 Scope of the Present Work

Many researchers worked on synthesis for non-redundant and planar redundant manipulators for specific tasks. But no work is reported for the synthesis of spatial redundant manipulators. The present work considers the problem of synthesis in spatial environment based upon techniques of optimization and a better tool for obstacle avoidance like proximity query package. In this work, type and number synthesis for manipulator is done before going for dimensional synthesis. If failure is reported for particular type and number of links then the process is repeated for dimensional synthesis again and again by different type and number of links. Results show good promise for synthesis of spatial redundant manipulator.

1.5 Organization

The next chapter briefly outlines the tools used in the thesis. Problem formulation and constraint handling is discussed in Chapter 3. Chapter 4 includes a set of examples to verify and evaluate the approach adopted. The work is concluded in Chapter 5 with a summary and future scope in the field. Flow charts for different codes developed and templates of input files needed for the codes are supplied in appendices, for easy reference.

Chapter 2

Optimization Methods and Obstacle Avoidance

In this chapter, Augmented Lagrangian method and Genetic Algorithm are introduced in brief. A brief introduction of Proximity Query Package (PQP) is also given which has been utilized for the purpose of obstacle avoidance followed by a brief outline of the methodology used to solve the present problem.

2.1 Optimization Methods

2.1.1 Augmented Lagrangian Method

The method was conceptualised by Powell, Hestenes and Rockafellar [25, 26, 27] based on combining duality with exterior penalty functions. The penalty function method is designed to converge by making penalty parameter (R) high, which results in very steeply curved functions and in turn causes poor rate of convergence, as in this situation Hessian of Lagrangian becomes very ill-conditioned [28]. On the other hand, duality concept in Augmented Lagrangian method introduces a strong assumption, that $\nabla^2\mathbf{L}$, i.e., Hessian of Lagrangian is locally convex at $\mathbf{x}^*$ but this assumption is quite justified in most of the localities. This assumption makes the convergence smooth and gradual. If $f(\mathbf{x})$ is the objective function to be minimized, subject to the inequality constraints

$$g_i(\mathbf{x}) \leq 0; \qquad i = 1, 2, \ldots, I;$$

and equality constraints

$$h_j(\mathbf{x}) = 0; \qquad j = 1, 2, \ldots, J;$$

where I is the number of inequality constraints and J is the number of equality constraints. The Augmented Lagrangian function to be optimized with equality and inequality constraints is given as

$$F(\mathbf{x}) = f(\mathbf{x}) + \sum_{j=1}^{J} \lambda_j h_j(\mathbf{x}) + \frac{1}{2} r \sum_{j=1}^{J} [h_j(\mathbf{x})]^2 + \sum_{i=1}^{I} \mu_i g_i(\mathbf{x}) + \frac{1}{2} r_1 \sum_{i=1}^{I} [\max(0, g_i(\mathbf{x}))]^2 \tag{2.1}$$

where λ_j and μ_i are the Lagrange multipliers for j-th and i-th equality and inequality constraints respectively and r and r_1 are the penalty parameters for violation of the same.

The scheme followed for the update of Lagrange multipliers for $(k+1)$-th iteration is given below. Update for an equality constraint is conducted as

$$\lambda_{j,k+1} = \lambda_{j,k} + r h_j(\mathbf{x}) \tag{2.2}$$

where λ_j can take any value positive, negative or zero.

For an inequality constraint, Lagrange multiplier μ_j is updated such that it remains either positive or zero, as

$$\mu_{j,k+1} = \begin{cases} \mu_{i,k} + r_1 g_i(\mathbf{x}) & \text{if } g_i(\mathbf{x}) \geq 0.0; \\ \mu_{i,k+1} = 0.0 & \text{otherwise;} \end{cases} \tag{2.3}$$

where k is the number of k-th Augmented Lagrangian iteration.

2.1.2 Genetic Algorithm

The concept of Genetic Algorithms was introduced by Holland [29] in 1975. Genetic Algorithms are population based search and optimization techniques. Some features like their ability to provide a robust serach in complex search spaces with discontinuities and ability to reach a global optimum in a complex search space [30, 31] makes them more suitable for this type of problems. Use of GAs also avoids evaluation of Jacobians and gradients, so that any problem related to non-analyticity is overcome automatically. The simple working cycle of GA is shown in the Fig. 2.1.

There exist several versions of GA, such as binary-coded GA, real-coded GA and others. Details of the particular method can be seen in references Goldberg [31] and Deb [32].

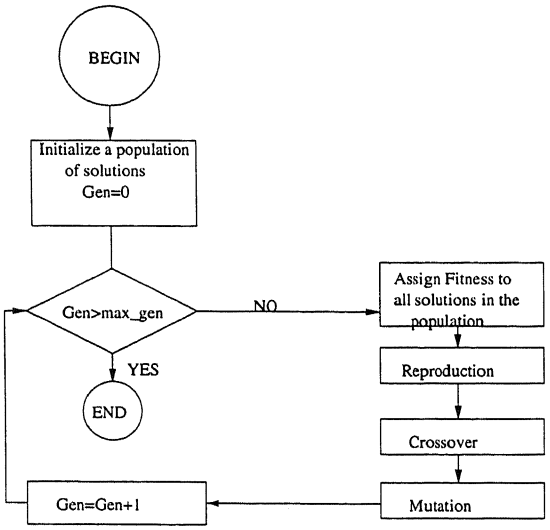


Figure 2.1: Schematic diagram of working cycle of GA

In binary-coded GA, all problem variables are coded in finite-length binary strings and fitness is evaluated based upon these. For maximization problems, a string's fitness can be equal to the objective function value. However for minimization problems, the string fitness can be related to the reciprocal of the objective function value as

$$Fitness = \frac{1}{[1 + f(\mathbf{x})]} \tag{2.4}$$

where $\mathbf{x}$ is the variable vector and $f(\mathbf{x})$ is the objective function.

For any GA, there are three important operators viz. reproduction, crossover and mutation. Reproduction operator selects good strings from the population using fitness value of the function and forms a mating pool. Using the most popular scheme of tournament selection out of many reproduction schemes discussed in Goldberg *et al.* [33], better string is being chosen by comparing two strings and copied in a mating pool. This process continues till the mating pool has the same size as the original population size initially taken.

In crossover, new strings are created by exchanging information among two good strings of the mating pool found after reproduction. There are several types of crossover techniques disscussed in Deb [32]. Crossover operator is mainly responsible for the search of a new strings.

In order to preserve already existing good strings crossover is performed with a probability called crossover probabilty (p_c) slightly less than one.

Mutation operator produces new string from an old existing string by altering it locally. This is responsible for sudden changes in the characteristic of some members of population. This change occurs at a very small probability value called mutation probability μ_c.

One generation of a GA completes when reproduction, crossover and mutation are applied to the whole population once. New strings are created by crossover and mutation. Out of these, good strings are emphasized and bad strings suppressed by means of reproduction operator.

In this work real-coded GA has been used. *Simulated Binary Crossover* (SBX) (refer [32]) technique has been used for crossover, and mutation is done using a randomly calculated perturbance parameter.

2.2 Proximity Query Package

In order to avoid the collision of manipulator links with obstacles as well as collision of one link with another, a software package called *Proximity Query Package* (PQP) is used. PQP is a library which uses swept sphere *bounding volumes* (BV) for *collision detection*, *separation distance computation* and *tolerance limit verification*. Gottschalk *et al.* [23, 24] presented this novel algorithm using swept sphere volumes for efficient and exact interference detection among complex models undergoing rigid body motion.

Many real life problems have stringent performance requirements. They need to conduct these queries in less than a millisecond on large models composed of hundereds or thousands of polygons. To obtain all these, many algorithms have been proposed based on different *bounding volume hierarchies* (BVH). Out of these, the most efficient is bounding volume hierarchy based on *swept sphere volumes*. Efficiency of a hierarchy is affected by the choice of a BV type as it depends upon two conflicting factors, *Tightness of fit* and *speed of operation* with BVs. So a trade-off is obtained between these.

A bounding volume is used to bound or contain sets of geometric primitives such as triangles, polygons etc. In this package a family of three BV's has been used for proximity queries.

Point Swept Sphere (PSS), Line Swept Sphere (LSS) and Rectangle Swept Sphere (RSS). The use of this family as BV's is justified because of advantages like efficiency, varying tightness of fit, ability to make hybrid combinations and low storage requirements.

Algorithms for building BVH's has two parts: enclosing a set of triangles by PSS, LSS or RSS depending upon suitability and grouping of nested BV's into a single hierarchy.

Given a set of triangles, statistical techniques are used to compute different bounding volumes. Using these statistical data, *oriented bounding box*(OBB) is computed which encloses the underlying geometry. Using the dimensions of OBB's different BV's, like PSS, LSS and RSS is computed. When the OBB has three axes of similar lengths, a PSS is used and when one axis length is relatively larger than the other two then LSS is the best choice, in all other case RSS is used. A top-down strategy is being used to create the node hierarchy. While building a BVH, BV's are stored at the internal nodes of a tree structure and the root BV contains all the primitives of a model and children BV's each contain separate partitions of the primitives enclosed by the parents. Leaf node BVs typically contain one primitive, and splitting rule used here is the same as used for an OBB tree, Fig. 2.2 (Gottschalk *et al.* [24]).

The algorithm used for proximity query utilizes the properties of *external Voronoi regions*

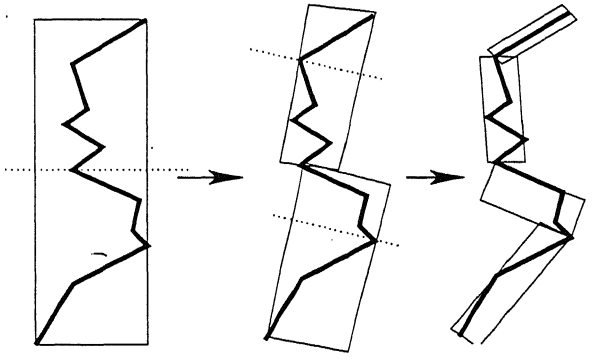


Figure 2.2: Building the OBB tree: recursively partition the bounded polygons and bound the resulting groups.

(Lin *et al.* [34]), which is algorithmcally stated in the following.

- Determine, If the closest points between the rectangles lie on the boundary edges

- IF YES, Compute the distance between these points and return it
- ELSE, One of the two closest points must lie in the interior of a rectangle

The three possible cases of voronoi region are shown in Fig. 2.3 (Gottschalk *et al.* [24]).

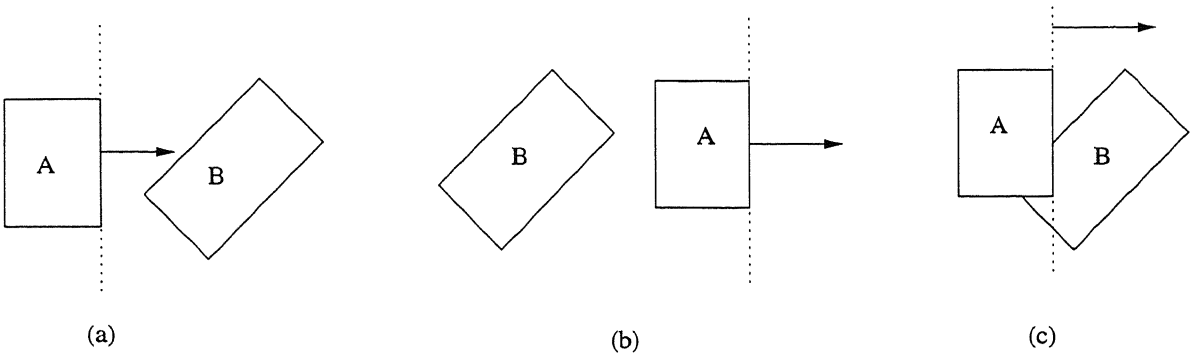


Figure 2.3: Different relative configuration of two edges of rectangles. (a) B is entirely inside Voronoi Region of A, (b) B is entirely outside of Voronoi region of A (c) Some points of B are overlapping with Voronoi region of A.

The minimum distance between two points is returned for the non-overlapping and overlapping cases, and for this the technique of variation of separating axis is used. The idea is to project each rectangle to a unit direction and compute the distance between resulting intervals. This distance along a direction is a lower bound to the actual distance Fig. 2.4. So, once the distances between rectangle-rectangle is calculated all others become special cases of this.

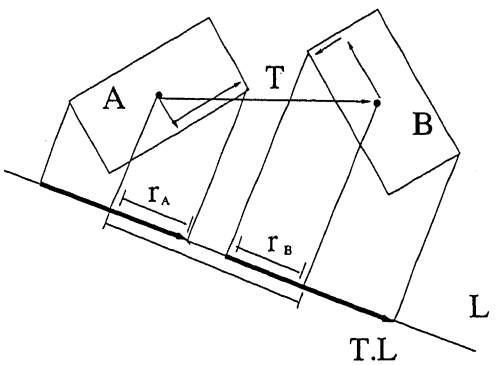


Figure 2.4: Variation of separation method

The distance computation by the above algorithm is utilized for collision detection and relative error tolerance measurement by means of *dilation of shapes* concept [23]. Further techniques of priority *directed search* and *triangle caching* is used to accelerate the whole process of PQP.

2.3 Probabilistic Roadmap Method for Path Planning

This is the tool to plan the path between two task space locations. The underlying idea is that, instead of searching the C-space[1] exhaustively, one tries to randomly select some configurations (milestones) in the C_{free} space and link them through a network called *roadmap*.

2.4 Methodology

The concepts discussed above facilitates the formulation and solution of the problem in forthcoming chapters. Methodology to solve the problem is given here in brief.

Step 1 Perform the type and number synthesis beforehand.

Step 2 Formulate the objective function.

Step 3 Formulate the constraints due to parameter bounds and obstacle avoidance.

Step 4 Apply the optimization methods to solve the problem subjected to constraints.

Step 5 If satisfactory solution is reached then terminate, otherwise, go to Step 1.

Here type and number synthesis is done before applying the present algorithm.

[1]The set of all possible configurations of the manipulator with reference to a fixed reference

Chapter 3

Formulation of the Synthesis Problem

The objective of the present work is *to synthesize a kinematically redundant spatial manipulator for reachability, avoiding obstacles.* The problem is formulated as an optimization problem with reachability as the objective with obstacle avoidance and bounds on D-H parameters[1] as constraints. The problem will be discussed with sufficient elaboration in the forthcoming sections.

3.1 Formulation of the Objective Function

The reachability[2] of the manipulator is considered as the objective function. For the present problem this is measured as the total cumulative error in **Euclidean distances** i.e., error incurred in the position of end-effector of manipulator to reach the desired TSL's while negotiating all the TSL's. A schematic diagram of a general redundant manipulator is shown in the Fig. 3.1.

To simplify the problem yet retaining sufficient extent of generality certain assumptions have been made, such as

1. The manipulator architecture is considered to be serial.

2. End-effector of the manipulator may have any orientation[3].

[1]Denavit-Hartenberg parameters

[2]Ability to reach all the specified TSL's by the manipulator end-effector.

[3]Explicitly it can be assumed that there is a wrist attached at the end-effector having three degree of freedom, yaw, pitch and roll which takes care of orientation.

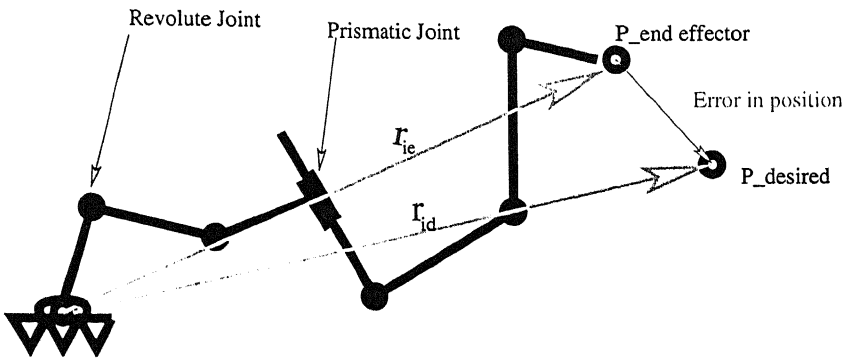


Figure 3.1: Schematic diagram of a general redundant manipulator.

3. Joints present are revolute/prismatic or combinations thereof[4].

Based on the above assumptions, if the manipulator has n number of links and there are N TSL's which have to be accessed by the manipulator end-effector, then if the position of the end-effector corresponding to the i-th TSL is defined by the vector $\mathbf{r}_{ie}$ and desired TSL is defined by vector $\mathbf{r}_{id}$ then square[5] of the error in the position, which is to be attained for the i-th TSL, is given as

$$P_{i_{error}} = \|\mathbf{r}_{id} - \mathbf{r}_{ie}\|^2 \tag{3.1}$$

In cartesian coordinates, Eq. 3.1 can be expressed as

$$P_{i_{error}} = (x_{id} - x_{ie})^2 + (y_{id} - y_{ie})^2 + (z_{id} - x_{ze})^2 \tag{3.2}$$

where (x_{id}, y_{id}, z_{id}) and (x_{ie}, y_{ie}, z_{ie}) are the Cartesian coordinates corresponding to points represented by the vectors $\mathbf{r}_{id}$ and $\mathbf{r}_{ie}$ respectively. Finally, total cumulative error for all the N TSL's can be written as

$$P_{error} = \sum_{i=1}^{N} (x_{id} - x_{ie})^2 + (y_{id} - y_{ie})^2 + (z_{id} - x_{ie})^2 \tag{3.3}$$

The TSL's desired to be reached by the end-effector are specified by the user, so coordinates of these points, i.e., (x_{id}, y_{id}, z_{id}) are known to the user. To find out Cartesian cordinates of

[4]This includes most other standard lower pair joints so for as 'geometry' is concerned.

[5]Instead of taking actual Euclidean distance i.e., square root of the $P_{i_{error}}$, the square error $P_{i_{error}}$ itself is taken, which causes no loss of generality.

position of end-effector in terms of variable D-H parameters, kinematic equations have to be developed. For this, first we fix coordinate frames at each joint of the manipulator according to the convention and find out the individual transformation matrices from one frame to the next. If the global coordinate frame of reference is numbered as "0", the frame at the end of end-effector by "n" and any frame in between is denoted by, say "i". The transformation matrix for i-th link i.e., from $(i-1)$-th frame to i-th frame, below is represented by T_i^{i-1}, Craig *et al.* [35].

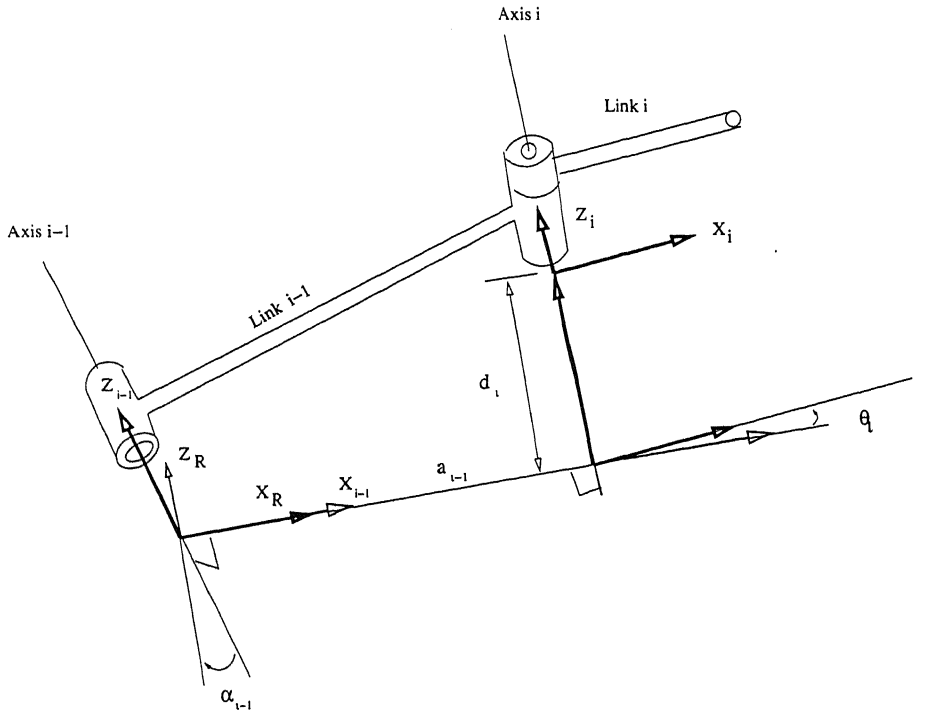


Figure 3.2: Schematic diagram of intermediate frames of link.

$$T_i^{i-1} = \begin{bmatrix} cos\theta_i & -sin\theta_i & 0 & a_{i-1} \\ sin\theta_i cos\alpha_{i-1} & cos\theta_i cos\alpha_{i-1} & -sin\alpha_{i-1} & -sin\alpha_{i-1} d_i \\ sin\theta_i sin\alpha_{i-1} & cos\theta_i sin\alpha_{i-1} & cos\alpha_{i-1} & cos\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

Here $[a_{i-1}, \alpha_{i-1}, d_i, \theta_i]$ are D-H parameters of i-th link of the manipulator. In case the base of manipulator is not at origin but is located at some point, say (x_b, y_b, z_b), then a translational

transformation given by the following should be applied beforehand, assuming that global frame of reference is at origin, i.e., $(0, 0, 0)$.

$$T_{translation} = \begin{bmatrix} 1 & 0 & 0 & -x_b \\ 0 & 1 & 0 & -y_b \\ 0 & 0 & 1 & -z_b \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

In this way, all the transformation matrices can be obtained and final transformation matrix for end-effector with respect to global frame of reference can be written as

$$T_n^0 = T_{translation} T_1^0 T_2^1 T_3^2 \cdots T_n^{n-1} \tag{3.6}$$

First three element of the fourth column of global transformation matrix (T_n^0) give coordinates of the end-effector i.e. (x_{ie}, y_{ie}, z_{ie}).

Finally the objective function P_{error} is formulated in terms of (x_{id}, y_{id}, z_{id}) and (x_{ie}, y_{ie}, z_{ie}) from Eq. 3.3. This P_{error} is to be minimized subject to the constraints discussed in the following section.

3.2 Handling Constraints

3.2.1 Constraints for Augmented Lagrangian Method

Constraint due to Boundary Values of D-H parameters

D-H parameters of the links have certain bounds. These bounds have been imposed as constraints for the optimization method. The bounds on D-H parameters for i-th link can be written as

$$a_{li} \leq a_i \leq a_{ui} \tag{3.7}$$

$$\alpha_{li} \leq \alpha_i \leq \alpha_{ui} \tag{3.8}$$

$$\begin{cases} d_{li} \leq d_i \leq d_{ui} \\ \text{and} \\ \theta_i = 0 \end{cases} \tag{3.9}$$

if i-th joint is **prismatic.**

$$\begin{cases} \theta_{li} \leq \theta_i \leq \theta_{ui} \\ \text{and} \\ d_i = 0 \end{cases} \tag{3.10}$$

if i-th joint is **revolute.**

where $i = 1, 2, \ldots, n$

Inequalities in Eqs. 3.7, 3.8, 3.9 and 3.10 are treated as inequality constraints and each inequality gives rise to a set of two inequality constraints,

$$\begin{cases} g_{i1}(\mathbf{x}) = a_{li} - a_i \leq 0; \\ g_{i2}(\mathbf{x}) = a_i - a_{ui} \leq 0; \end{cases} \tag{3.11}$$

Similarly other inequalities above can be written as inequality constraints.

Equality constraints are due to the equalities of Eqs. 3.9 and 3.10, and are written as

$$\begin{cases} h_i(\mathbf{x}) = \theta_i = 0; \\ \text{or} \\ h_i(\mathbf{x}) = d_i = 0; \end{cases} \tag{3.12}$$

depending upon the i-th joint being **prismatic** or **revolute** respectively. $\mathbf{x}$ is the vector of design variables i.e., set of D-H parameters[6], n is the number of joints and N is the number of TSL's.

The above mentioned vector can be split into two parts, one only constituted of (a, α, d) and other of all θ only. This gives the option to find optimum set of geometric parameters for any set of joint variable θ.

Output of PQP

The PQP package gives three result once the triangles of the manipulator and obstacles are sent for check. These are

- **Minimum Distance**: If there is no collision then it returns minimum distance between any pair of triangles, else zero.
- **Collision Check**: If there is collision then it returns the pair of triangles between which collision occurs.

[6] $[a_0, a_1, \ldots, a_{n-1}; \alpha_0, \alpha_1, \ldots, \alpha_{n-1}; d_1, d_2, \ldots, d_n; \theta_{11}, \theta_{12}, \ldots, \theta_{1n}; \theta_{ki}, \ldots, \theta_{Nn}]$

- **Tolerance Check**: It returns the binary value for the question, "Is there any pair of triangles between the specified tolerance limit ?"

Constraints due to obstacles

First two results are the basis of obstacle avoidance, and the following algorithm is used based on the above results to avoid obstacles

- **IF** there is collision, then the extent of collision is calculated for each pair of colliding triangles of manipulator and obstacle (for self-intersection, triangles of one link with other links is considered but not links adjacent to each other[7]) and distance of penetration is returned (**negative** in nature). Suppose it is designated as **distP** .

- **ELSE** returns the minimum distance of separation (**positive** in nature). Suppose it is also denoted by **distP**.

The inequality constraint including both checks is written as

$$g_i(\mathbf{x}) = -distP \leq 0 \tag{3.13}$$

where $i = 1, \ldots, N$ and N is the number of TSL's.

Constraints due to decreasing links length towards end-effector

This constraint is not mandatory, but can be taken from operational viewpoint. It is assumed that the length of i-th link is greater than or equal to length of $(i+1)$-th link i.e., $(a_{i+1} \leq a_i)$. This constraint can be written as a set of inequality constraints as follows:

$$g_i(\mathbf{x}) = a_{i+1} - a_i \leq 0 \qquad i = 1, 2, \ldots, (n-1) \tag{3.14}$$

where n is number of links.

Thus Eqs. 3.11, 3.13 & 3.14 give all the inequality constraints and Eq. 3.12 is the set of

[7]The collision between two adjacent links can be avoided by judiciously chosing the angle limits to save computational cost.

all equality constraints. Using the above constraints of equality and inequality, the final unconstrained function of Augmented Lagrangian can be written as given in Eq. 2.1. This function is minimized using Polak-Ribiere[8] method of unconstrained optimization in the minimization loop of the Augmented Lagrangian method.

3.2.2 Constraint Handling in Genetic Algorithm

The penalty form used is simple penalty as proposed in Deb [32]. In case of Genetic Algorithms the bounds on variables are supplied as limits on the variables given in Eqs. (3.7–3.10). For obstacle avoidance, if there is any collision, penetration distance is calculated as discussed above and is used to penalize the objective function to be minimized. Penalty term is calculated depending upon the total penetration summed up over all the pairs in contact or intersection. Expression for penalty term corresponding to each TSL is given as:

$$P_{penalty} = \begin{cases} Ae^{-B*distP} & \text{if} \quad distP \leq 0 \\ 0 & \text{Otherwise;} \end{cases} \tag{3.15}$$

where A and B are user defined penalty parameters.

The constraint due to link lengths can also be utilized in the same fashion as for obstacle avoidance by putting some fixed penalty value for each constraint violation and adding them after multiplying with square of constraint violation to the original objective function. In the same fashion as in Eq. 3.14, constraints can be written as

$$g_i(\mathbf{x}) = a_i - a_{i+1} \geq 0 \qquad i = 1, 2, \ldots, (n-1); \tag{3.16}$$

so the final function value for Genetic algorithm is evaluated by adding the objective function with penalty term written as

$$F(\mathbf{x}) = f + P_{penalty} + \sum_{i=1}^{n-1} P\left[\max\{-g_i(\mathbf{x}), 0\}\right]^2 \tag{3.17}$$

where P is fixed penalty value defined by the user. Final fitness value is evaluated using the function value as in Eq. 2.4. New fitness function is given as

$$Fitness = \frac{1}{[1 + F(\mathbf{x})]} \tag{3.18}$$

[8]This is advantageous over Fletcher-Reeves method because it is useful for functions, even if they are not exactly quadratic in nature.

Chapter 4

Results and Discussion

The formulations of the last chapter are coded into C++ programs. The method of optimization used are Augmented Lagrangian and real-coded Genetic Algorithm. The results and inferences obtained are shown and discussed in this chapter. The D-H parameters obtained for different manipulators in increasing order of complexity of work cell after synthesis are put in tables. There are the following three kinds of plots for different manipulators and complexities,

1 **Schematic diagram of Manipulator configurations in workspace.**

2 **Schematic diagram of paths between two task-space locations in the workspace.**

3 **Plot of minimum distance of any point of manipulator from any point of the obstacle vs. milestone numbers in the path.**

These results depend upon the input parameter values for **Augmented Lagrangian** and **Genetic Algorithms** summarized in each case. In each example discussed here, the manipulator base is assumed to be at $(0, 0, 0)$, which causes no loss of generality because the base point of robot can be considered as reference point. With respect to this, all other reference frames can be assigned. There is also provision made in the **"code"** internally so that base of manipulator can be chosen anywhere depending upon the will of the user. One can do it by making a simple modification in the input file[1]. The format of input files needed for the code is attached in the appendix. The link cross-section for simplicity is assumed to be square and half-width of that

[1]specify the base coordinates (x_b, y_b, z_b) as non-zero

is usually taken as ($\frac{1}{20}$ to $\frac{1}{10}$) times the maximum limit on links length. The units of any length measurement are in **centimeters** and angles are measured in **radians**. System time reported for each run is **actual time** taken for one complete run of program instead of actual CPU time.

4.1 Example 1

Consider the problem of design of a manipulator which has 5 links with four revolute and one cylindrical joints for a given workspace. The workspace has 3 obstacles blocks scattered in space (see Fig. 4.1).

Separate listing of input parameter values for **Augmented Lagrangian** and **Genetic Algorithm** is given here. D-H parameters obtained after synthesis for listed input parameter values is shown in Table 4.1.

Parameter Values for Augmented Lagrangian Method	
Half width of link(t)	0.1
No. of links	5
No. of prismatic joints	1
Position of prismatic joint	2
No of destination points	2
Destination points TSL's.	(4,4,4) & (7,5,5)
Bounds on link length	(0 3)
Bounds on twist angle	(-1 1) (radians)
Bounds on offset value at prismatic joints	(-1 1)
Bounds on θ values	(-3 3)
Maximum iterations for Augmented Lagrangian loop	100
Penalty Parameter Value ($r = r_1$)	5000.0

Parameter Value for Genetic Algorithm	
Half width of link(t)	0.1
No. of links	5
No. of prismatic joints	1
Position of prismatic joint	2
No. of destination points	2
Destination points TSL's.	(4,4,4) & (7,5,5)
Max no. of generation	2000
Population size	500
Crossover probability	0.9
Mutation probability	0.1
No. of design variables	25
Type of GA used (real or binary)	r
Lower and upper limits on variables	(0 3), (-1 1), (-1 1) & (-3 3)
Are these bounds rigid ? (y/n)	y
Sharing is to be done ? (y/n)	y
If yes, sharing parameter	0.1
How many runs?	1
Selection operator	Tournament selection
Enter tournament size	2
Type of crossover stratgey	Crossover in all variables
Type of crossover(s/b)	Simulated Binary Crossover
η value for Simulated Binary X and mutation	10 100
Random seed no.	0.123

Table 4.1: Table of D-H parameters for 5 link manipulator, *Aug. Lag. run*

S. N.	Link Length (a_{i-1})	Twist (α_{i-1})	Offset (d_i)	Aug. Lag. execution statistics.	GA execution statistics
1	2.173689	-0.177532	0.918337	time	time
2	1.835387	0.806187	0.000000	taken=	taken=
3	1.908823	0.620726	0.000000	39 (min.)	831.21(min.)
4	2.100628	0.918821	-0.000000	Accuracy	Accu.
5	2.262606	0.916501	0.000000	10^{-6}	10^{-2}

4.1.1 Discussion

Example 4.1 is the simplest case considered in the present work. The manipulator is requireed to reach two TSL's specified as in the input parameter table. Fig. 4.1 shows the configuration corresponding to two TSL's with one cylindrical joint at the joint number one. It is clear

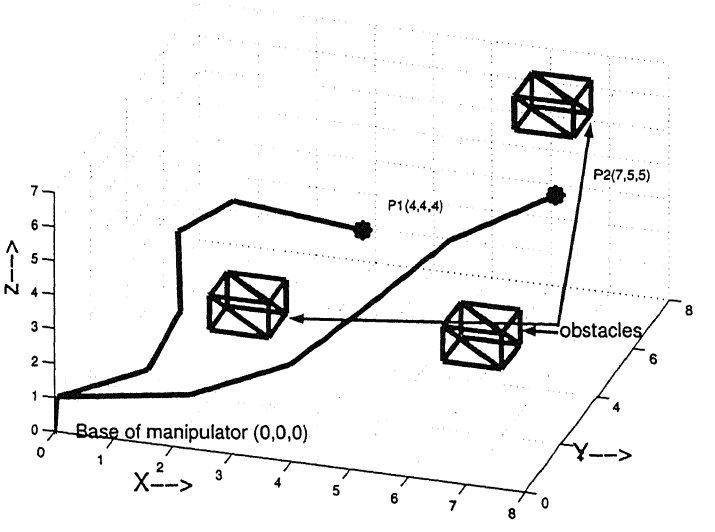


Figure 4.1: 5 link manipulator configuration with 3 obstacle blocks.

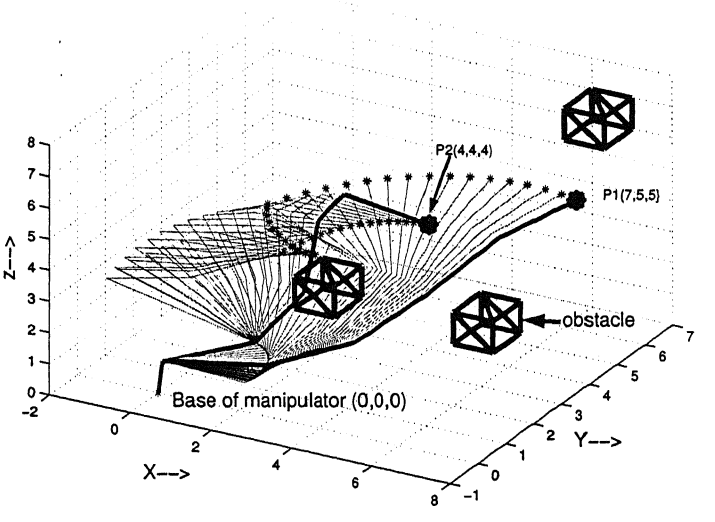


Figure 4.2: Schematic diagram of path between two TSL's.

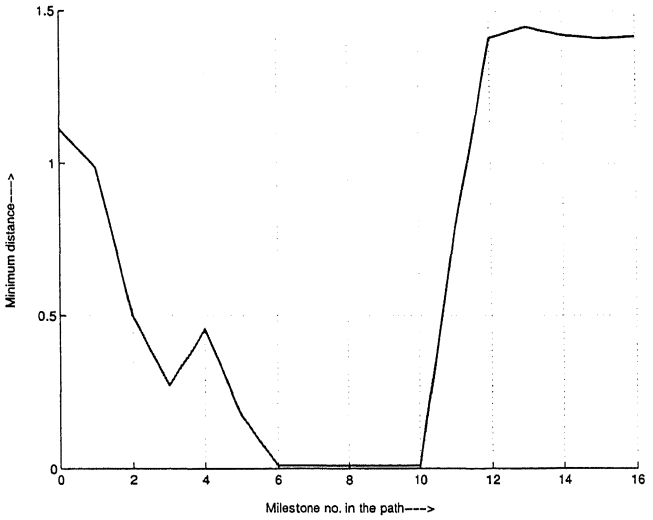


Figure 4.3: Plot of minimum distance of any point of obstacle from any point of manipulator *vs* corresponding milestone no. between points P1 and P2.

from the figure that the manipulator is reaching the TSL's while avoiding the obstacles. The problem considered in this example is simple as there are only three cubical obstacle blocks present, as shown in the figure, and these obstacles are also very simple in geometry. Fig. 4.2 shows a feasible path in which milestones present in the path are represented by "*" along with intermediate link configurations[2]. From the figure, it is not clearly evident that the manipulator avoids the obstacle. It is because of the difficulty delineating spatial figures in planar one. For this case, the **minimum distance** of any point on manipulator link in any particular configuration from any point on the obstacles present in the workspace is obtained from PQP (Section 2.2) and plotted against the corresponding milestone number present in the path. This plot is shown in Fig. 4.3. From this figure it is clear that when the manipulator is in configuration corresponding to milestone numbers 6, 7 and 8, it is very close to the obstacle but avoiding it with very small tolerance value of 0.01 units.

Here, a little explanation will be appropriate to avoid any confusion regarding the obstacle avoidance. Since, while formulating the problem it is considered, that in case there is any

[2]Path is found using the code of Probabilistic Road Map and is not the optimum. This is only a feasible path

collision, then the minimum distacne returned will be negative irrespective of the magnitude. Hence, if there is collision then path finding routine itself avoids that configurationn and there is no question of getting an infeasible path.

In this example one extra degree of freedom is present at joint number one due to presence of combined revolute and prismatic joints. Considering all this, it comes out to be total 3 degree of redundancy which can be calculated as

```
Total no. of degree of freedom = Total DOF present in the manipulator
Total no. of degree of freedom = 5+1=6
    Total degree of redundancy = Total DOF present - minimum DOF needed
                               = 6-3 = 3
```

The results obtained from Augmented Lagrangian and Genetic Algorithms are very much same and so only one result table is shown. The major difference is the time taken in executing the program as shown in result Table-4.1. The accuracy also in the case of Genetic Algorithms is poor whereas in case of Augmented Lagrangian it is very good. This is due to the limitation of simple method adopted for constraint handling in case of Genetic Algorithm.

4.2 Example 2

In this example, problem is to design a manipulator which has 6 links and with first joint as cylindrical for workspaces with 3 and 5 cubical obstacle blocks scattered in space.

D-H parameter obtained after synthesis with listed input parameter values is given in Tables 4.2 and 4.3. Different results based upon these data are shown in figures 4.4, 4.5, 4.6, 4.7 and 4.8.

Parameter Values for Augmented Lagrangian Method	
Half width of link(t)	0.1
No. of links	6
No. of prismatic joints	1
Position of prismatic joint	1
No. of destination points	3
Destination points TSL's.	(4,4,4), (6,8,5), (7,5,5)
Bounds on link length	(0 3)
Bounds on twist angle	(-1 1) (radians)
Bounds on offset value at prismatic joints	(-1 1)
Bounds on θ values	(-3 3)
Maximum. iterations for Augmented Lagrangian loop	10
Penalty Parameter Value ($r = r_1$)	5000.0

Parameter Value for Genetic Algorithm	
Half width of link(t)	0.1
No. of links	6
No. of prismatic joints	1
Position of prismatic joint	1
No. of destination points	3
Destination points TSL's.	(4,4,4), (6,8,5), (7,5,5)
Max no. of generation	5000
Population size	540
Crossover probability	0.9
Mutation probability	0.1
No. of design variables	36
Type of GA used (real or binary)	r
Lower and upper limits on variables	(0 3), (-1 1), (-1 1) & (-3 3)
Are these bounds rigid ? (y/n)	y
Sharing is to be done ? (y/n)	y
If yes, sharing parameter	0.1
How many runs?	1
Selection operator	Tournament selection
Enter tournament size	2
Type of crossover stratgey	Crossover in all variables
Type of crossover(s/b)	Simulated Binary Crossover
η value for Simulated Binary X and mutation	10 100
Random seed no.	0.123

Table 4.2: Table of D-H parameters for 6 link manipulator, *Aug. Lag. run for 3 obstacle*

S. N.	Link Length (a_{i-1})	Twist (α_{i-1})	Offset (d_i)	Aug. Lag. Exe. Statistics	GA Exe. Statistics
1	2.401201	-0.286105	0.673107	Time=	Time=
2	1.864476	0.813343	0.000001	17.520717	950.34
3	1.289989	0.121528	0.000000	(min.)	(min)
4	1.795214	0.728001	0.000001	Accu.=	Accu.
5	2.449541	0.776908	0.000001	10^{-6}	10^{-3}
6	2.346068	0.890235	0.000001		

Table 4.3: Table of D-H parameters for 6 link manipulator, *Aug. Lag. run for 5 obstacle*

S. N.	Link Length (a_{i-1})	Twist (α_{i-1})	Offset (d_i)	Aug. Lag. Exe. statistics	GA Exe. statistics
1	2.404906	-0.282519	0.673332	Time=	Time=
2	1.871103	0.816416	0.000000	29.639550	954.34
3	1.300629	0.118279	-0.000000	(min.)	(min)
4	1.802409	0.716956	0.000000	Accuracy	Accuracy
5	2.451770	0.766542	-0.000000	10^{-6}	10^{-3}
6	2.347554	0.885253	-0.000000		

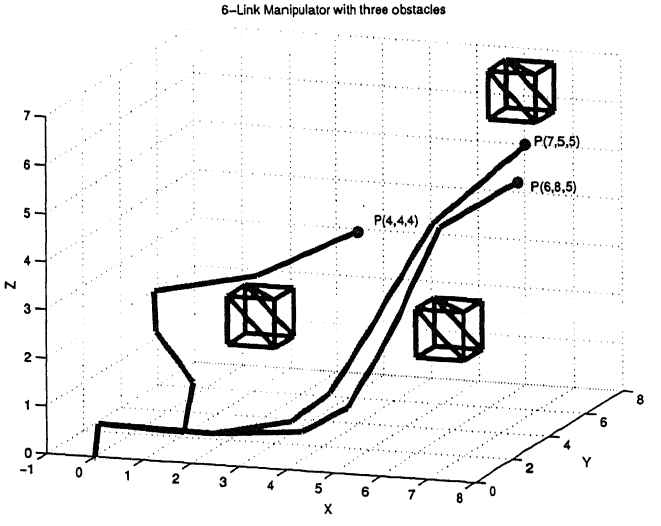


Figure 4.4: 6 link manipulator configuration with 3 obstacle blocks.

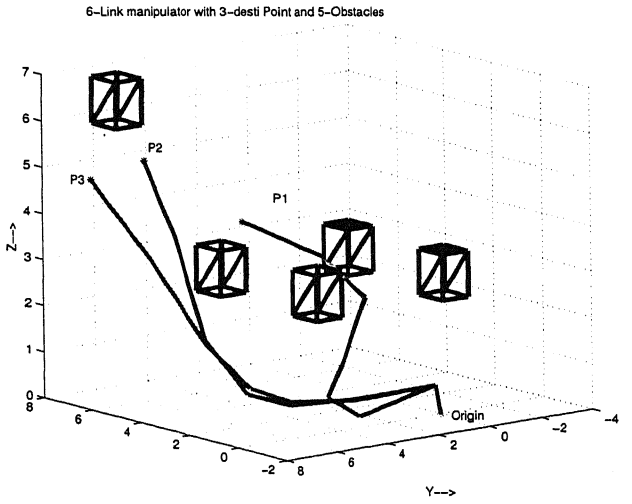


Figure 4.5: 6 link manipulator configuration with 5 obstacle blocks.

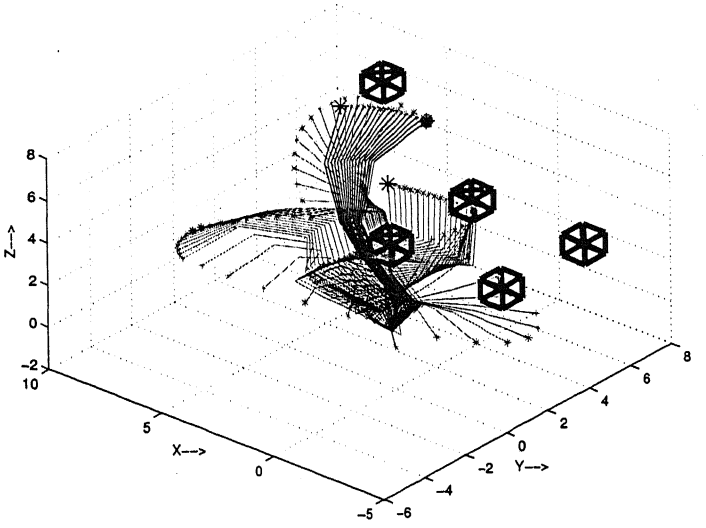


Figure 4.6: Schematic diagram of path between two TSL's.

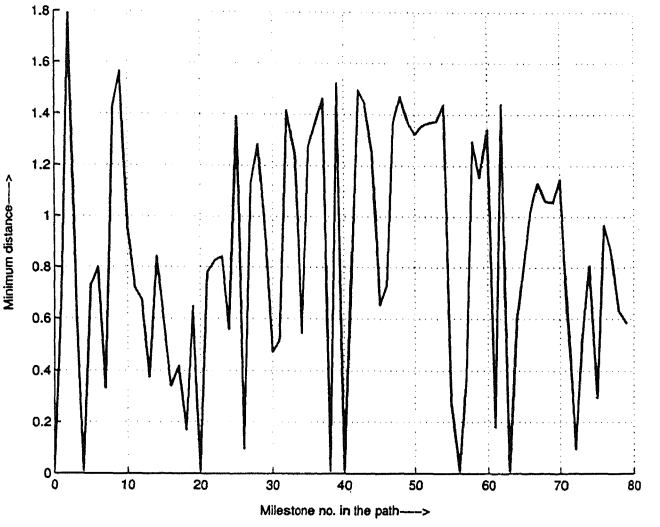


Figure 4.7: Plot of minimum distance of any point of obstacle from any point of manipulator *vs.* corresponding milestone no. while negotiating the path between points P1 and P2.

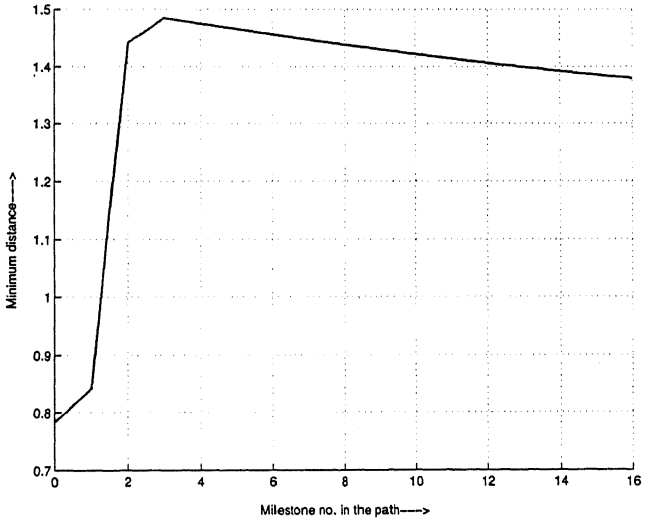


Figure 4.8: Plot of minimum distance of any point of obstacle from any point of manipulator *vs.* milestone no. while negotiating the path between points P2 and P3.

4.2.1 Discussion

The configurations of manipulator in two different workspaces are shown in Fig. 4.4 and Fig. 4.5. These results are corresponding to D-H parameter values given in Tables 4.2 and 4.3 respectively based on Augmented Lagrangian and GA methods. There are three TSL's which the manipulator has to reach in each case. The path between these TSL's with intermediate link configuration for the latter case only is shown in Fig. 4.6. The plot of minimum distance is also shown only for the second case in Fig. 4.7 and Fig. 4.8, respectively for paths between points P1–P2 and P2–P3.

In this case also, the major difference between Augmented Lagrangian and Genetic Algorithm comes in with respect to time of excecution of program and accuracy attained.

The total **degree of redundancy** available is **four**[3].

4.3 Example 3

Design a manipulator which has 8 links without any prismatic joint for a given workspace. The workspace has cubical blocks as obstacles as shown in Fig. 4.9.

D-H parameter values obtained after synthesis with listed input parameters are shown in Table 4.4. Plot of configurations of manipulator at different TSL's, path between two TSL's and minimum distance plot of manipulator from obstacle vs. milestone number in the path is shown in Figs. 4.9, 4.10, 4.11, 4.12.

[3]see subsection 4.1.1 for calcualtion of DOR

Parameter Values for Augmented Lagrangian Method	
Half width of link(t)	0.1
No. of links	8
No. of prismatic joints	0
Position of prismatic joint	no value set
No. of destination points	3
Destination points TSL's	(4,-4,3),(6,1,0),(1,5,0)
Bounds on link length	(0 3)
Bounds on twist angle	(-1 1) (radians)
Bounds on offset value at prismatic joints	(-1 1)
Bounds on θ values	(-3 3)
Maximum. iterations for Augmented Lagrangian loop	10
Penalty Parameter Value ($r = r_1$)	5000.0

Parameter Value for Genetic Algorithm	
Half width of link(t)	0.1
No. of links	8
No. of prismatic joints	0
Position of prismatic joint	no value set
No. of destination points	3
Destination points TSL's.	(4,-4,3),(6,1,0),(1,5,0)
Max no. of generation	8000
Population size	720
Crossover probability	0.9
Mutation probability	0.1
No. of design variables	48
Type of GA used (real or binary)	r
Lower and upper limits on variables	(0 3), (-1 1),(-1 1) and (-3 3)
Are these bounds rigid ? (y/n)	y
Sharing is to be done ? (y/n)	y
If yes, sharing parameter	0.12
How many runs?	1
Selection operator	Tournament selection
Enter tournament size	2
Type of crossover stratgey	Crossover in all variables
Type of crossover(s/b)	Simulated Binary Crossover
η value for Simulated Binary X and mutation	10 100
Random seed no.	0.123

Table 4.4: Table of D-H parameters for 8 link manipulator, *Aug. Lag. run*

S. N.	Link Length (a_{i-1})	Twist (α_{i-1})	Offset (d_i)	Aug. Lag. Exe. Statistics	GA Exe. Statistics
1	2.614835	-0.148001	0.000000	time	time
2	2.470839	0.867388	0.000000	3.4672	1543.31
3	1.847867	-0.535727	0.000000	(min)	(min)
4	1.799706	-0.214662	0.000000		
5	1.595607	-0.615296	0.000000		
6	1.531483	0.100523	0.000000	Accuracy	Accuracy
7	0.899789	-0.883619	0.000000	10^{-5}	10^{-3}
8	0.704201	0.398194	0.000000		

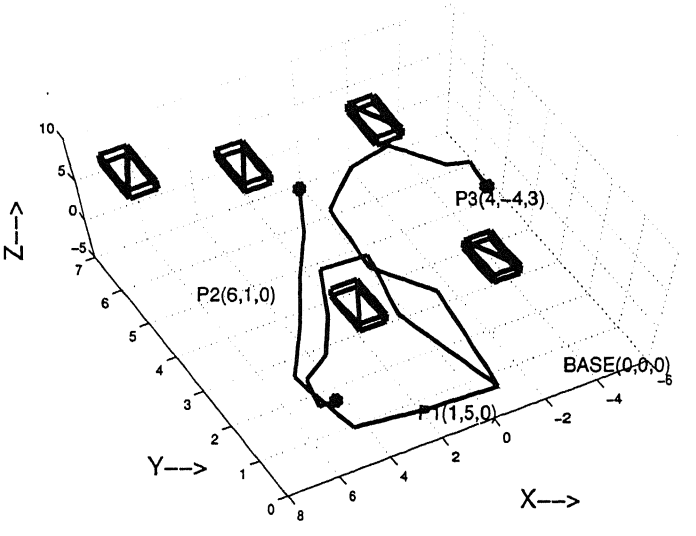


Figure 4.9: 8 link manipulator configuration with 5 obstacle blocks.

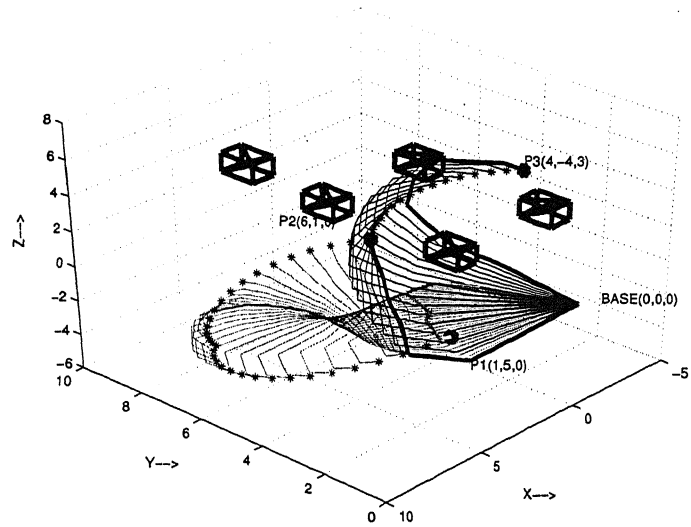


Figure 4.10: Schematic diagram of path between two TSL's.

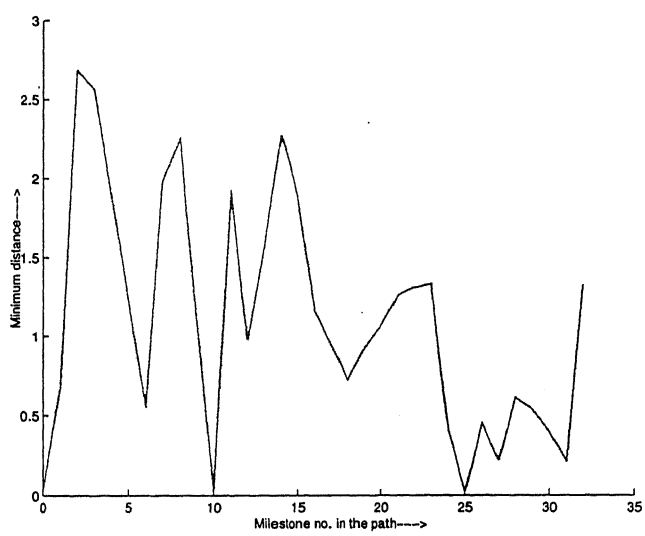


Figure 4.11: Plot of minimum distance of any point of obstacle from any point of manipulator *vs.* corresponding milestone no. while negotiating the path between points P1 and P2.

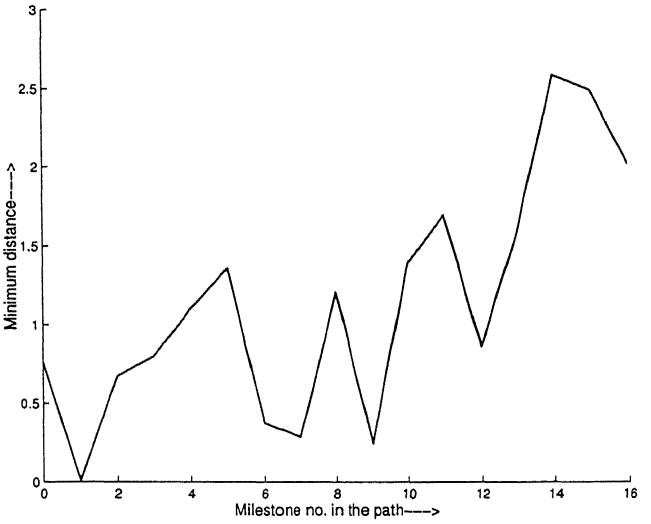


Figure 4.12: Plot of minimum distance of any point of obstacle from any point of manipulator *vs.* milestone no. while negotiating the path between points P2 and P3.

4.3.1 Discussion

The results have been plotted as for the previous examples. Here plots of configuration, path and minimum distance all are similar as shown in Figs 4.9, 4.10, 4.11 and 4.12 respectively. From the plot of minimum distance, i.e., Fig. 4.11, the manipulator is found very near to the obstacle but is still avoiding the obstacle by the specified tolerance limit[4] of 0.01 unit.

The results obtained from Augmented Lagrangain is similar to Genetic Algorithms method and only one result table is shown and different plots have been obtained only from the results of this table. The Table 4.4 gives a good comparison of time of excecution of the program and accuracy obtained for the same complexity of problem on the same P-IV computing machine. The **degrees of redundancy** available here is five as no joint in this case is cylindrical.

4.4 Example 4

Building up on the previous examples, let us consider the synthesis of a manipulator which has 10 links with first joint as cylindrical. The workspace has 10 cubical blocks scattered in the

[4]user-defined while finding path either by swelling the obstacle or manipulator.

space.

D-H parameter obtained after synthesis with listed input parameter values respectively from Genetic Algorithm and Augmented Lagrangian methods for two different cases is shown in Tables 4.5 and 4.6.

Parameter Values for Augmented Lagrangian Method	
Half width of link(t)	0.1
No. of links	10
No. of prismatic joints	1
Position of prismatic joint	1
No. of destination points	3
Destination points TSL's.	(8 7 3), (8 -7 3), (-9 0 6)
Bounds on link length	(0 3)
Bounds on twist angle	(-1 1) (radians)
Bounds on offset value at prismatic joints	(-1 1)
Bounds on θ values	(-3 3)
Maximum. iterations for Augmented Lagrangian loop	10
Penalty Parameter Value ($r = r_1$)	5000.0

Parameter Value for Genetic Algorithm	
Half width of link(t)	0.1
No. of links	10
No. of prismatic joints	1
Position of prismatic joint	2
No. of destination points	3
Destination points TSL's.	(7,5,5), (5,1,0), (4,-4,3)
Max no. of generation	8000
Population size	600
Crossover probability	0.9
Mutation probability	0.1
No. of design variables	60
Type of GA used (real or binary)	r
Lower and upper limits on variables	(0 3), (-1 1), (-1 1) and (-2 2)
Are these bounds rigid ? (y/n)	y
Sharing is to be done ? (y/n)	y
If yes, sharing parameter	0.5
How many runs?	1
Selection operator	Tournament selection
Enter tournament size	2
Type of crossover stratgey	Crossover in all variables
Type of crossover(s/b)	Simulated Binary Crossover
η value for Simulated Binary X and mutation	10 100
Random seed no.	0.123

Table 4.5: Table of D-H parameters for 10 link manipulator, *Aug. Lag run*

S. N.	Link Length (a_{i-1})	Twist (α_{i-1})	Offset (d_i)	Aug. Lag. Exe. Statistics	GA Exe. Statistics
1	1.706159	0.281337	1.454124	Time.	
2	1.706203	-0.769435	0.000054	3.8477	see
3	1.706286	-0.205340	0.000057	(min.)	table
4	1.706367	0.786767	-0.000004		4.6
5	1.706422	0.241471	-0.000018		.
6	1.706476	0.164566	-0.000019		
7	1.706542	0.543024	-0.000013	Accuracy	
8	1.706604	0.642626	-0.000006	10^{-4}	
9	1.706628	0.463014	-0.000010		
10	1.706645	0.239135	-0.000016		

Table 4.6: Table of D-H parameters for 10 link manipulator, *GA run*

S. N.	Link Length (a_{i-1})	Twist (α_{i-1})	Offset (d_i)	Aug. Lag. Exe. statistics	GA Exe. statistics
1	2.678147	0.477924	0.000000		Time
2	2.553220	-0.612307	-0.137431		3205.12
3	2.429541	-0.306620	0.000000		(min.)
4	2.346381	0.087631	0.000000	see	
5	2.251404	-0.851019	0.000000	table	Gen. No.
6	2.190497	0.119769	0.000000	4.5	7807
7	2.058086	-0.617751	0.000000		
8	1.669358	-0.540360	0.000000		Accuracy
9	1.411845	-0.934824	0.000000		10^{-3}
10	1.303153	-0.565571	0.000000		

4.4.1 Discussion

The results have been plotted for two different cases in the above example obtained from Augmented Lagrangian and Genetic Algorithms as well. Here configuration plot, plot of path

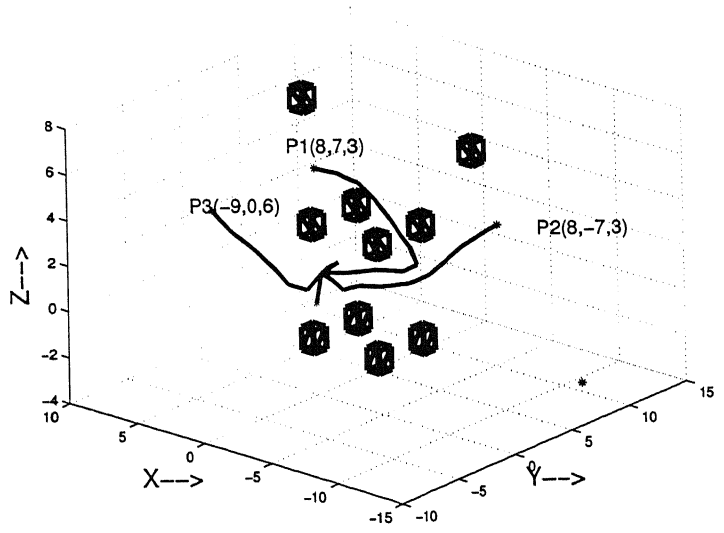


Figure 4.13: 10 link manipulator configuration obtained from *Aug. Lag. run.*

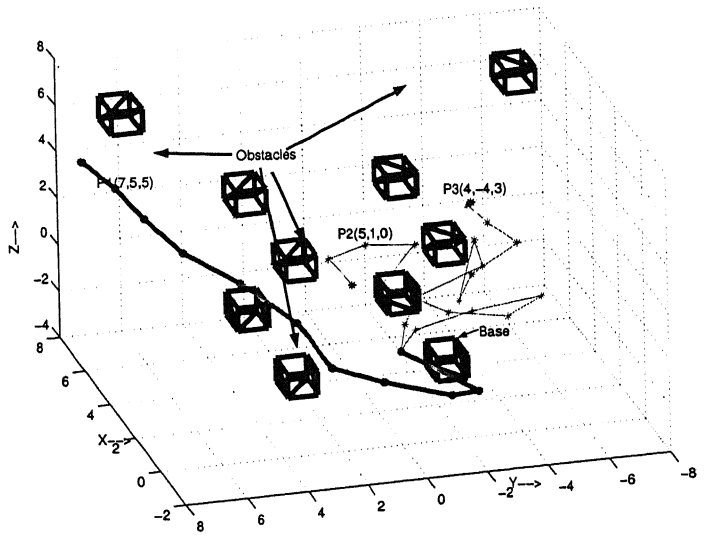


Figure 4.14: 10 link manipulator configuration with 10 obstacle blocks at point P1 *GA run.*

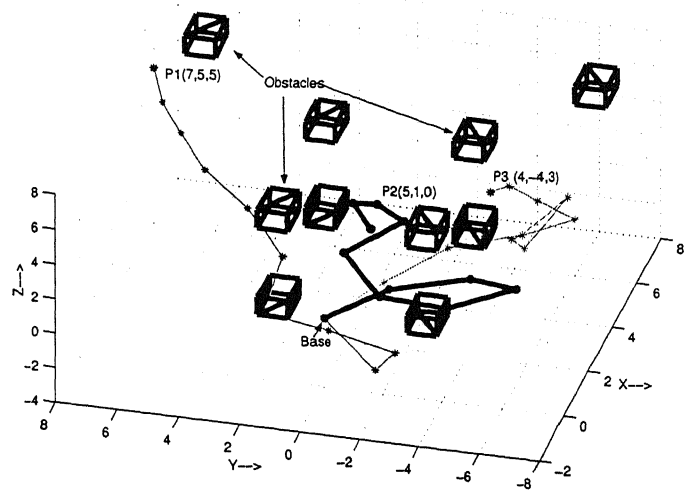


Figure 4.15: 10 link manipulator configuration with 10 obstacle blocks at point P2 *GA run.*

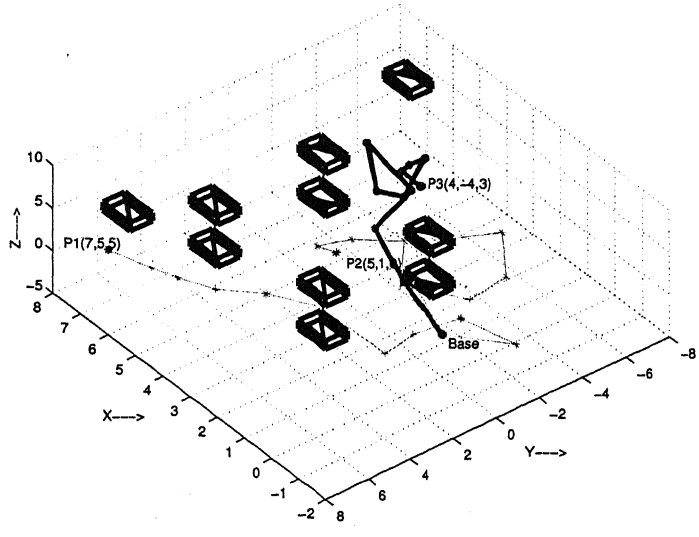


Figure 4.16: 10 link manipulator configuration with 10 obstacle blocks at point P3 *GA run.*

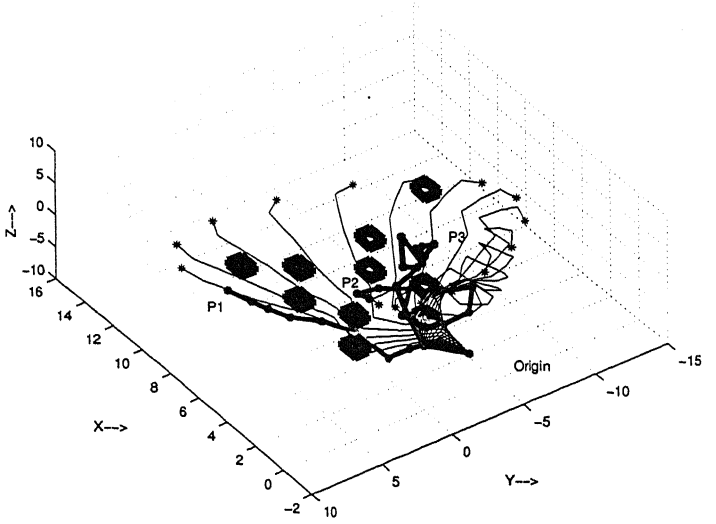


Figure 4.17: Schematic diagram of path between two TSL's i.e., points P1 and P2 shown along with link configurations.

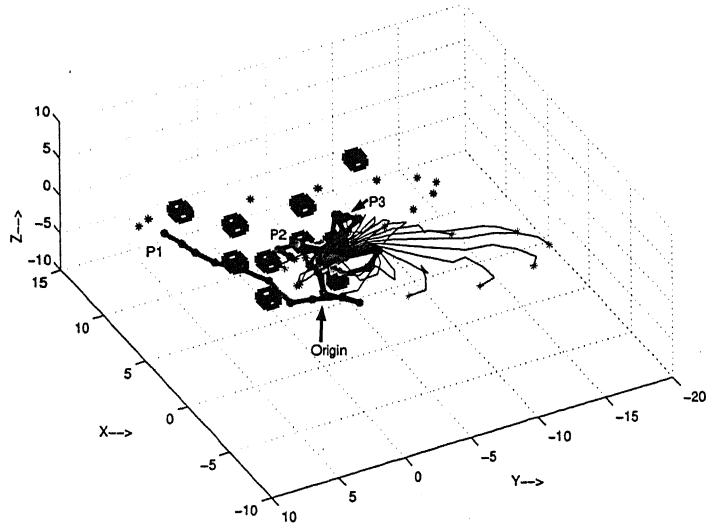


Figure 4.18: Schematic diagram of path between two TSL's i.e., points P2 and P3 shown along with link configurations.

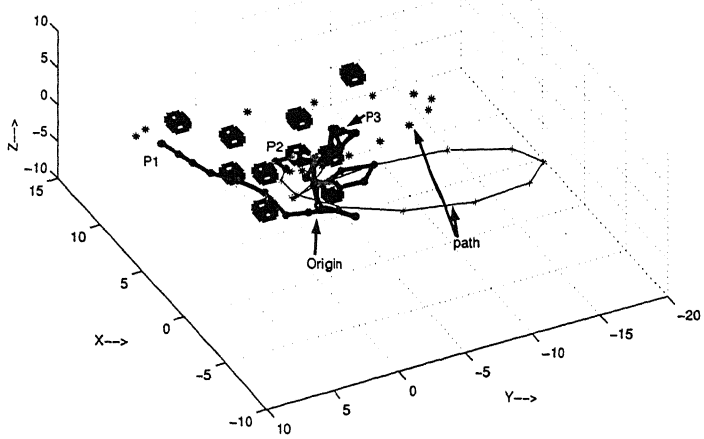


Figure 4.19: Schematic diagram of path between TSL's P1, P2 and P3.

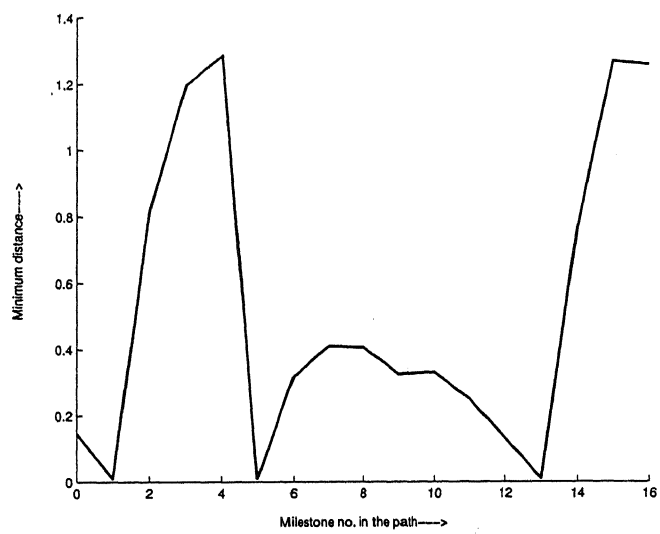


Figure 4.20: Plot of minimum distance of any point of obstacle from any point of manipulator *vs.* corresponding milestones no. while negotiating the path between points P1 and P2.

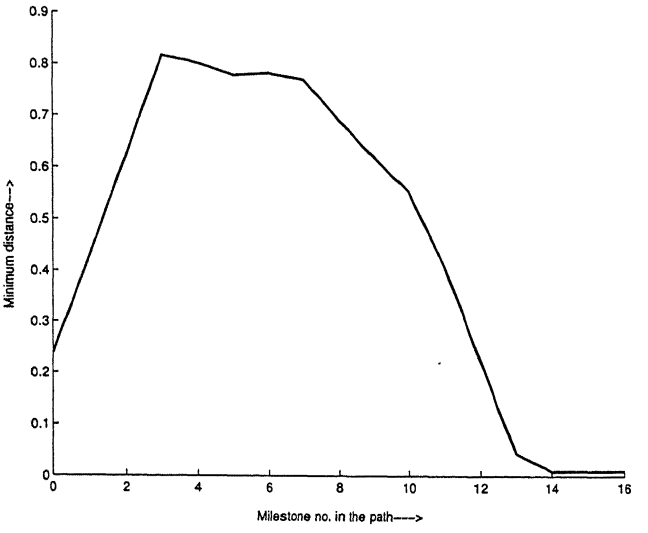


Figure 4.21: Plot of minimum distance of any point of obstacle from any point of manipulator *vs.* corresponding milestones no. while negotiating the path between points P2 and P3.

and minimum distance all are similar to the earlier examples. The plots are shown in Figs. 4.13, 4.14, 4.15, 4.16, 4.17, 4.18, 4.20 and 4.21.

The comparison of results obtained from Augmented Lagrangian and Genetic Algorithm respectively can be visualized by the inspection of results of Tables 4.5 and 4.6. The result clearly indicates that the accuracy obtained and time taken by the same complexity of problem (actually same problem) and on same computing machine is profoundly better in the case of Augmented Lagrangian method than the results obtained from Genetic Algorithms. The **degree of redundancy** avaibale is eight.

4.5 Example 5

Above examples give the impression that whatever be the number of links, solution is possible in every case. In this example, it is shown that in some cases it may not be possible to reach the TSL with prescribed number of DOF. For example, let us consider design of a manipulator for a cubicle workspace, with the below mentioned specifications

Parameter Values for Augmented Lagrangian Method	
Half width of link(t)	0.1
No. of links	8
No. of prismatic joints	1
Position of prismatic joint	2
No. of destination points	3
Destination points TSL's.	(6,8,5), (4,0,0),(4,-4,3)
Lower bound on link length (for each link)	0
Upper bounds on link length(for individual)	3
Bounds on twist angle	(-1 1) (radians)
Bounds on permanent offset at each joint	(-1 1)
Bounds on θ values	(-3 3)
Max. iterations for Augmented Lagrangian loop	100
Penalty Parameter Value ($r = r_1$)	5.0

Table 4.7: Table of D-H parameters for 8 link manipulator, *Aug. Lag run*

S. N.	Link Length (a_{i-1})	Twist (α_{i-1})	Offset (d_i)	Aug. Lag. Exe. statistics	GA Exe. statistics
1	1.0633	0.9300	0.3634	time	time
2	1.0160	0.6573	-0.5105	taken	taken
3	0.9348	0.3791	-0.5444	9.405	-NA-
4	0.8054	0.1351	-0.5960	min.	
5	0.6452	-0.0524	-0.6600		
6	0.4757	-0.1774	-0.7319	function	
7	0.3008	-0.2457	-0.8086	value	
8	0.1388	-0.2703	-0.8874	160.26	

4.5.1 Discussion

Form the Table 4.7 it is clear that the function value is far away from the desired (zero error) value. The constraint violation can be seen from the third column of Table 4.7. This clearly implies that the number of degree of freedom available is not sufficient for reaching the desired TSL, within the specified parameter bounds. In the next example workspace is same but number of DOF is increased. It is observed that the manipulator is able to reach the desired TSL within same parameter bounds.

4.6 Example 6

The same workspace with 10 links is now considered which has one cylindrical joint. The D-H parameters are obtained from synthesis using listed input parameter values. The results obtained are nearly same for both the optimization methods. This shows that the increased degree of freedom gives an solution for the same workspace in above example with same paramter bounds. Results are shown in the Table 4.8 and from this data, diffrent plots have been produced.

Parameter value for Genetic Algorithm	
Half width of link(t)	0.1
No. of links	10
No. of prismatic joints	1
Position of prismatic joint	2
No. of destination points	3
Destination points TSL's.	(6,8,5), (4,0,0), (4,-4,3)
Max no. of generation	8000
Population size	900
Crossover probability	0.9
Mutation probability	0.1
No. of design variables	60
Type of GA used (real or binary)	r
Lower and upper limits on variables	(0 3), (-1 1), (-1 1) and (-3 3)
Are these bounds rigid ? (y/n)	y
Sharing is to be done ? (y/n)	y
If yes sharing parameter	0.12
How many runs?	1
Selection operator	Tournament selection
Enter tournament size	2
Type of crossover stratgey	Crossover in all variables
Type of crossover(s/b)	Simulated Binary Crossover
η value for Simulated Binary X and mutation	10 100
Random seed no.	0.123

4.6.1 Discussion

It can be seen from the Fig. 4.22 that manipulator reaches the TSL which is inside the cubicle from the space available below the base with the 10 links.

Table 4.8: Table of D-H parameters for 10-link manipulator with a cubicle as obstacle having no base or roof in the workspace *GA run*

S. N.	Link Length (a_{i-1})	Twist (α_{i-1})	Offset (d_i)	Aug. Lag. Exe. statistics	GA Exe. statistics
1	2.217991	-0.488836	0.000000	Time=	Time=
2	2.165253	0.983744	-0.191289	107.78	1607.31
3	1.875893	0.230307	0.000000	(min.)	(min.)
4	1.865223	-0.617773	0.000000		
5	1.632666	0.873673	0.000000		No. of
6	1.515174	-0.455354	0.000000		Generation
7	1.484982	0.062152	0.000000	Accuracy	2592
8	1.213097	0.881003	0.000000	10^{-4}	
9	0.818835	0.452990	0.000000		Accuracy
10	0.812528	0.035390	0.000000		10^{-3}

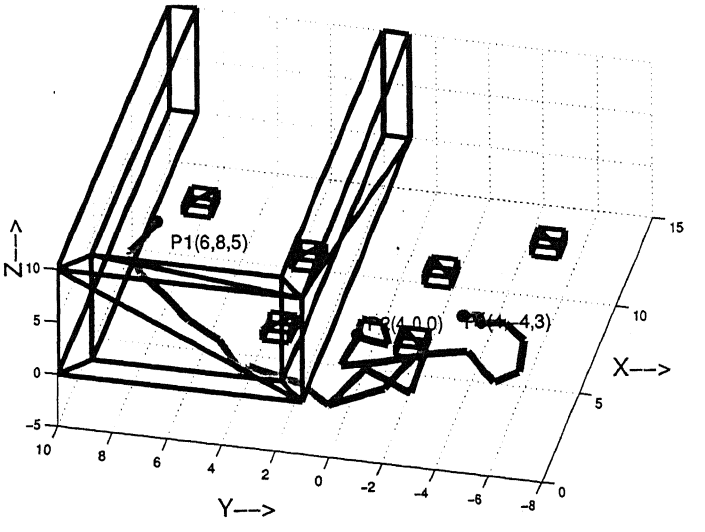


Figure 4.22: 10 link manipulator configuration with cubicle type workspace which have neither the base nor the roof.

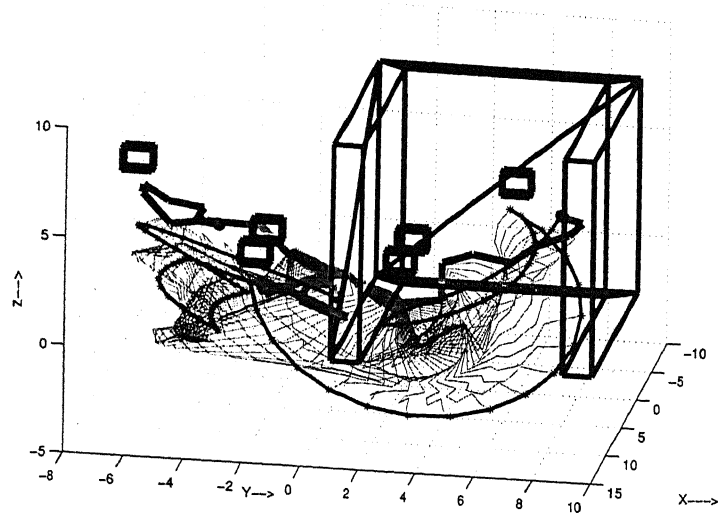


Figure 4.23: Schematic diagram of path between two TSLs i.e., point P1 and P2.

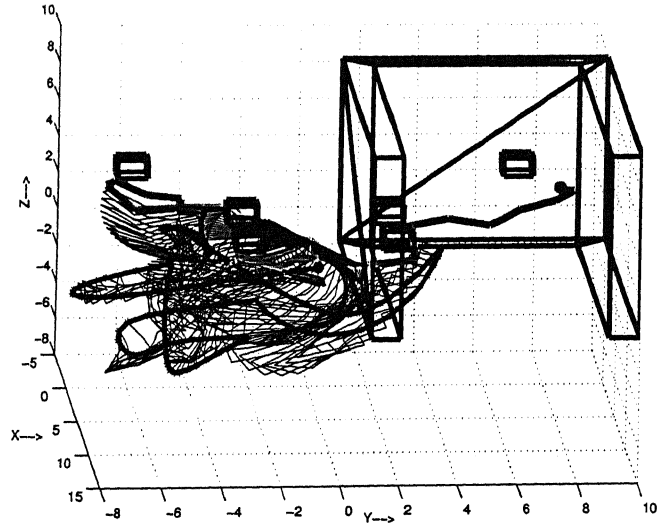


Figure 4.24: Schematic diagram of path between two TSLs i.e., point P2 and P3.

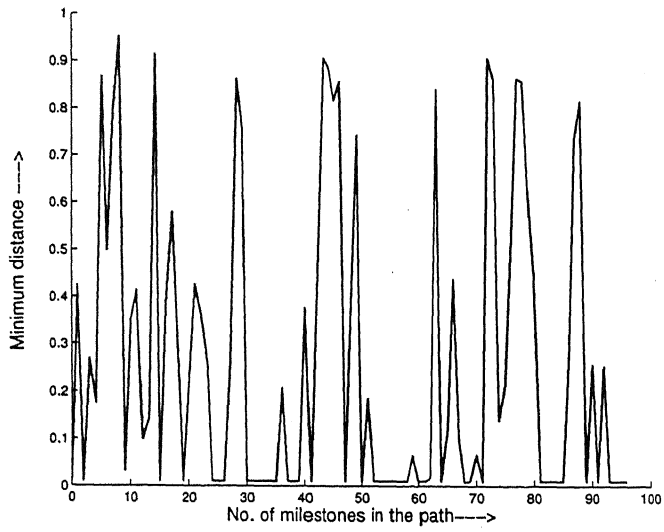


Figure 4.25: Plot of minimum distance of any point of obstacle from any point of manipulator *vs.* corresponding milestone no. while negotiating the path between points P1 and P2.

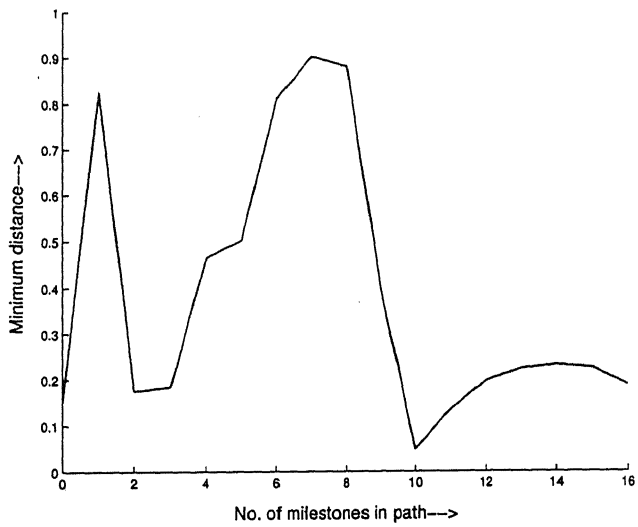


Figure 4.26: Plot of minimum distance of any point of obstacle from any point of manipulator *vs.* corresponding milestones no. while negotiating the path between points P1 and P2.

4.7 Example 7

This example is one step ahead with respect to complexity of workspace. Design a manipulator which has 14 links with one cylindrical joint. The synthesis results for the following listed input values have been produced in Table 4.9 which is obtained from Augmented Lagaranigan run. The results obtained from Genetic Algortim run are also similar except time and accuracy for the same problem. Workspace with manipulator configurations is depicted by the Figs. 4.27 and 4.28.

Parameter Values for Augmented Lagrangian Method	
Half width of link(t)	0.1
No. of links	14
No. of prismatic joints	1
Position of prismatic joint	1
No. of destination points	2
Destination points TSL's.	(-30,0,0), (0,30,0)
Bounds on link length	(0 3)
Bounds on twist angle	(-1 1) (radians)
Bounds on offset value at prismatic joints	(0 5)
Bounds on θ values	(-3 3)
Maximum. iterations for Augmented Lagrangian loop	15
Penalty Parameter Value ($r = r_1$)	500.0

Parameter Value for Genetic Algorithm	
Half width of link(t)	0.1
No. of links	10
No. of prismatic joints	1
Position of prismatic joint	2
No. of destination points	3
Destination points TSL's.	(-30,0,0), (0,30,0)
Max no. of generation	8000
Population size	1050
Crossover probability	0.9
Mutation probability	0.1
No. of design variables	70
Type of GA used (real or binary)	r
Lower and upper limits on variables	(0 3), (-1 1), (0 5) & (-3 3)
Are these bounds rigid ? (y/n)	y
Sharing is to be done ? (y/n)	y
If yes, sharing parameter	0.12
How many runs?	1
Selection operator	Tournament selection
Enter tournament size	2
w5 Type of crossover stratgey	Crossover in all variables
Type of crossover(s/b)	Simulated Binary Crossover
η value for Simulated Binary X and mutation	10 100
Random seed no.	0.123

4.7.1 Discussion

The workspace shown in this example is a bit complex, the manipulator end-effector has to reach the mid-point of the axial length of cylindrical holes present in the obstacle blocks. It is visible from Fig. 4.29 that the path obtained from PRM is very complex. Showing plots of link configurations for the shown path with "*" makes it worth nothing. The plot of minimum distance versus corresponding number of milestones tells the actual story that often some parts of the manipulator are very near to the obstacle.

Table 4.9: Table of D-H parameters for 14 link manipulator, *Aug. Lag run*

S. N.	Link Length (a_{i-1})	Twist (α_{i-1})	Offset (d_i)	Aug. Lag. Exe. statistics	GA Exe. statistics
1	2.999711	-0.374974	1.001334	time	time
2	2.999470	-0.561229	0.000000	taken	taken
3	2.936008	0.741706	0.000000	37.11	4135.28
4	2.701862	0.810150	0.000000	min.	min
5	2.701667	0.999615	0.000000		
6	2.664111	0.956880	0.000000		
7	2.664014	0.903536	0.000000		
8	2.594763	0.877791	0.000000		
9	2.582054	0.816532	0.000000	Accuracy	Accuracy
10	2.457285	0.939038	0.000000	10^{-6}	10^{-2}
11	2.403414	0.995549	0.000000		
12	2.269941	0.948659	0.000000		
13	2.269941	0.938980	0.000000		
14	2.216193	0.947669	0.000000		

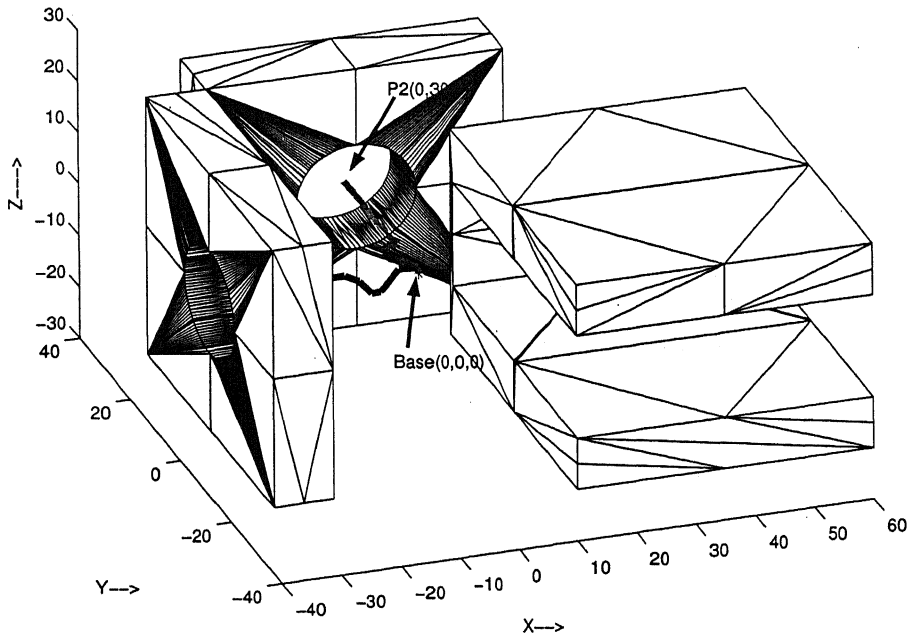


Figure 4.27: Spatial view of the manipulator in workspace showing link at P2

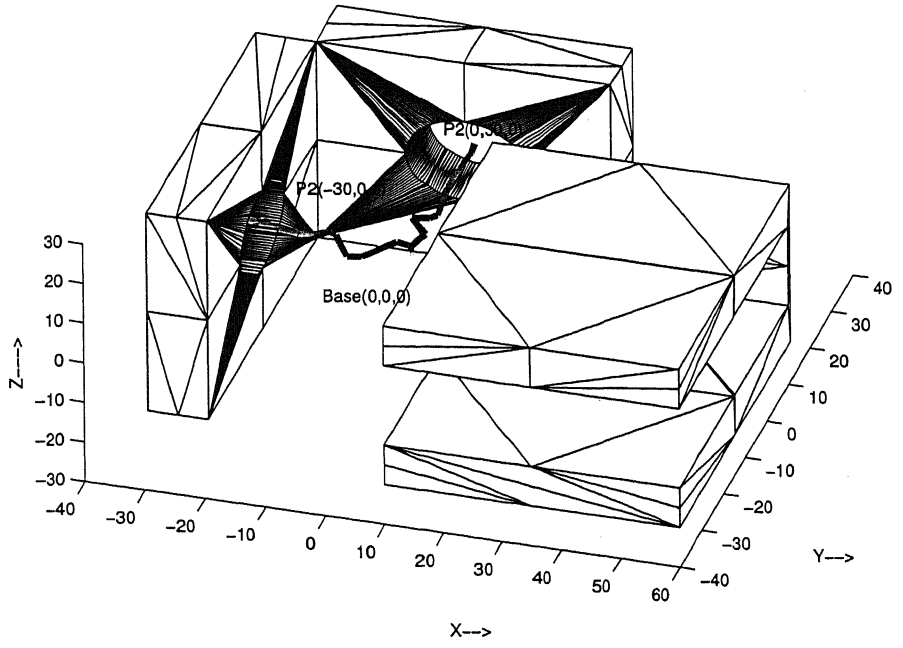


Figure 4.28: Spatial view of the manipulator in workspace showing link at position P1 and P2

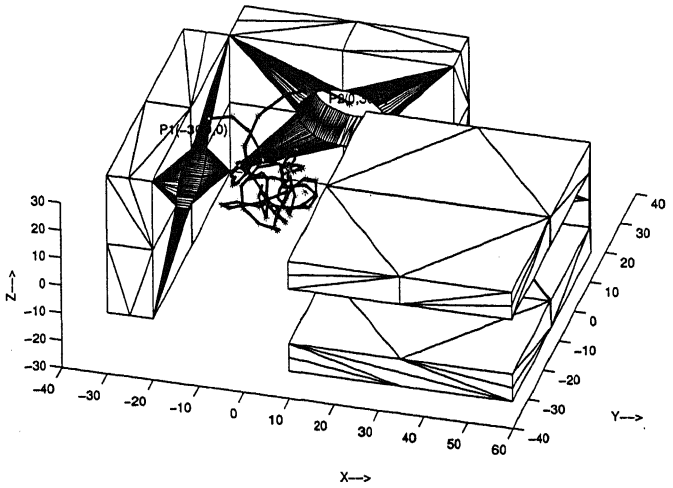


Figure 4.29: Path line in the workspace between points P1 and P2

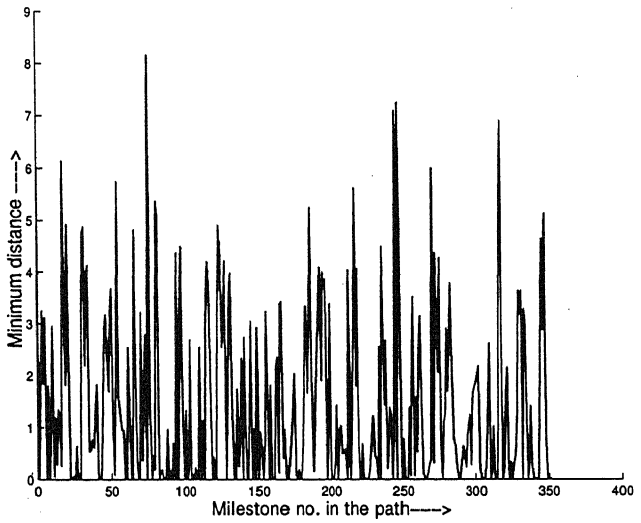


Figure 4.30: Plot of minimum distance between manipulator and obstacle *vs.* corresponding milestone no. between while travelling between points P1 and P2.

Chapter 5

Conclusions

5.1 Summary

The method discussed in the present work is a step ahead towards the work done in the field of synthesis of redundant manipulators. The algorithm with PQP to avoid the obstacle inherits the capability of avoiding any obstacle, whether it is convex or non-convex, quickly and efficiently.

The optimization problem in this method is solved using two methods, Augmented Lagrangian and Genetic Algortihms. The major conclusions which can be drawn are summarized as

1. The formulation is simple, general and capable of solving problems in various scenerios.
2. The formulation is independent of optimization methods to a great extent, except time and accuracy.
3. The formulation gives a feassible solution out of infinite solutions.
4. This method can be used for most standard lower pair joints, like cylindrical, spherical etc. with simple modification so far as 'geometry' is concerned.

In a nutshell, the method followed in this work gives a very general procedure for the synthesis of spatial redundant manipulators independent of the optimization methods used in general, but apparently time taken and accuracy depends upon the optimization methods used and associated parameters. So the process of optimization can be improved upon by chosing suitable

parameter values used and the methods of optimization depending upon the nature of problem in hand.

5.2 Future Scope

The success of the present work gives an option for exploring new aspects of synthesis problem which are mentioned below.

1. Mechanical design aspects of the manipulator can be considered, in which one can incorporate shape and size of links in the optimization procedure itself.

2. One can automate the type and number synthesis by incorporating it into the outer loop of optimization.

3. Many parameters defining qualitative nature of manipulator like manipulability, dexterity and isotropy can be included to make the problem complete.

4. One can explore the possibility of optimal and suboptimal designs.

5. The synthesis results can be utilized in practical design of manipulator.

Appendix A

Augmented Lagrangian Method

A.1 Dependencies of Modules in Augmented Lagrangian Method

main	auglagmn	readdata			
auglagmn	Func	dfunc	frprmn	updatelambda	updatemu
Func	constraintg	funct			
dfunc	Func				
funct	Eucled	collisioncheck			
collisioncheck	calpendist	PQP routines			
calpendist	calculatenormal				
calculatenormal					
Eucled					
frprmn	Func	dfunc	linmin		
linmin	mnbrak	brent			

A.2 Flow Chart of the Augmented Lagrangian Based Code

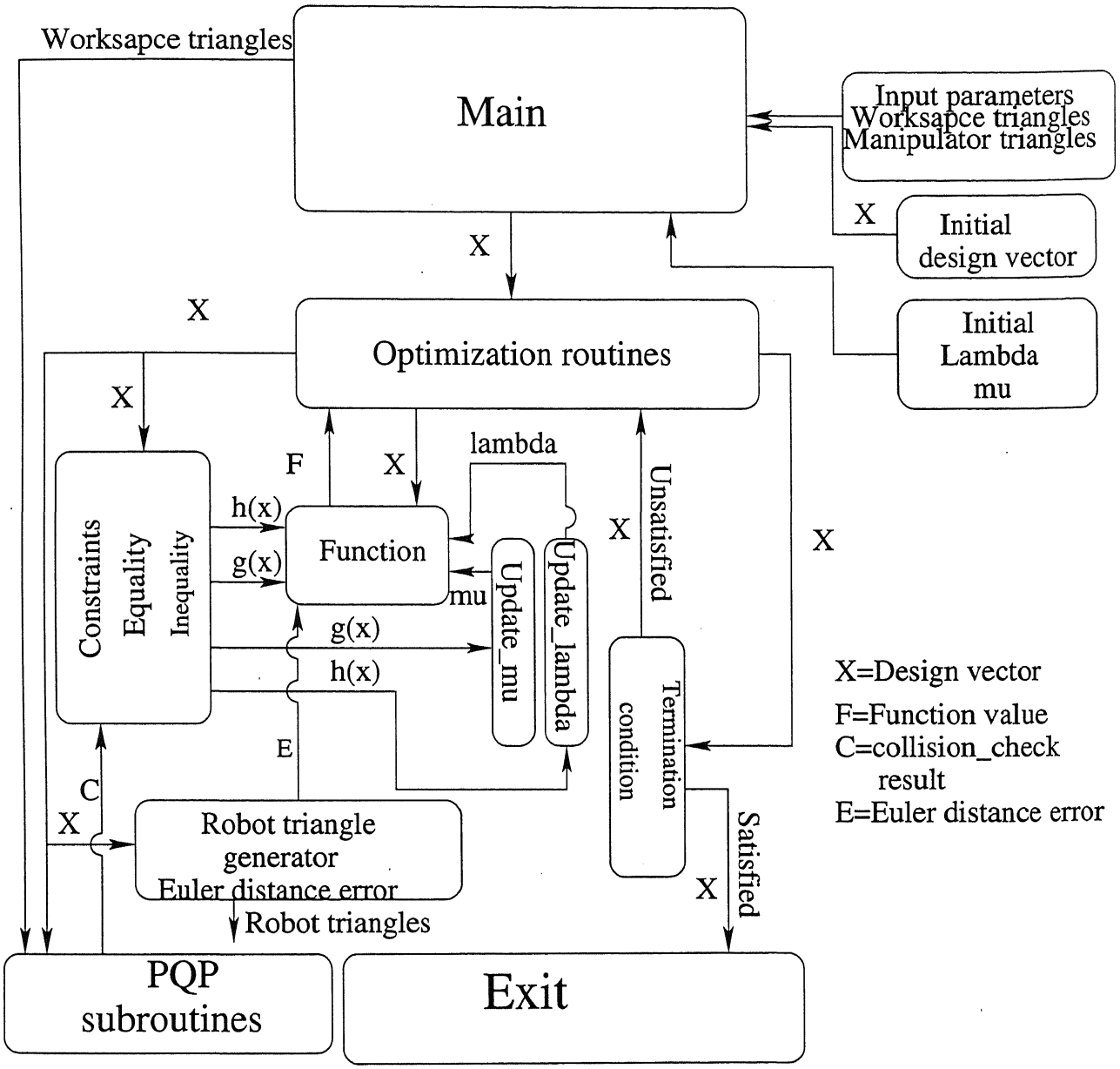


Figure A.1: Flowchart of the Augmented Lagrangian based code

A.3 Template of Input File for Augmented Lagrangian Based Code

Half width of manipulator links(tt) :(1/20 to 1/10)

Manipulator base coordinate :xb0 yb0 zb0

No. of links in the manipulator :nl

No. of TSL's :N

No. of prismatic joint :np

Position of prismatic joints :tem[]

Upper limit on link length (a) :lu

Lower limit on twist angle (α) :α_l

Upper limit on twist angle (α) :α_u

Lower limit on offset (d) :d_l

Upper limit on offset (d) :d_u

Lower limit on joint variable (θ) :θ_l

Upper limit on joint variable (θ) :θ_u

Appendix B

Genetic Algorithm Method

B.1 Dependencies of Modules in Genetic Algorithm Method

main	readdata	GACODE
GACODE	objective	
objective	funct	
funct	Eucled	collisioncheck
collisioncheck	calculatepenetrationdistance	PQP routines
calculatepenetrationdistance	calculatenormal	
calculatenormal		
Eucled		

B.2 Flow Chart of Real-coded Genetic Algorithm Based Code

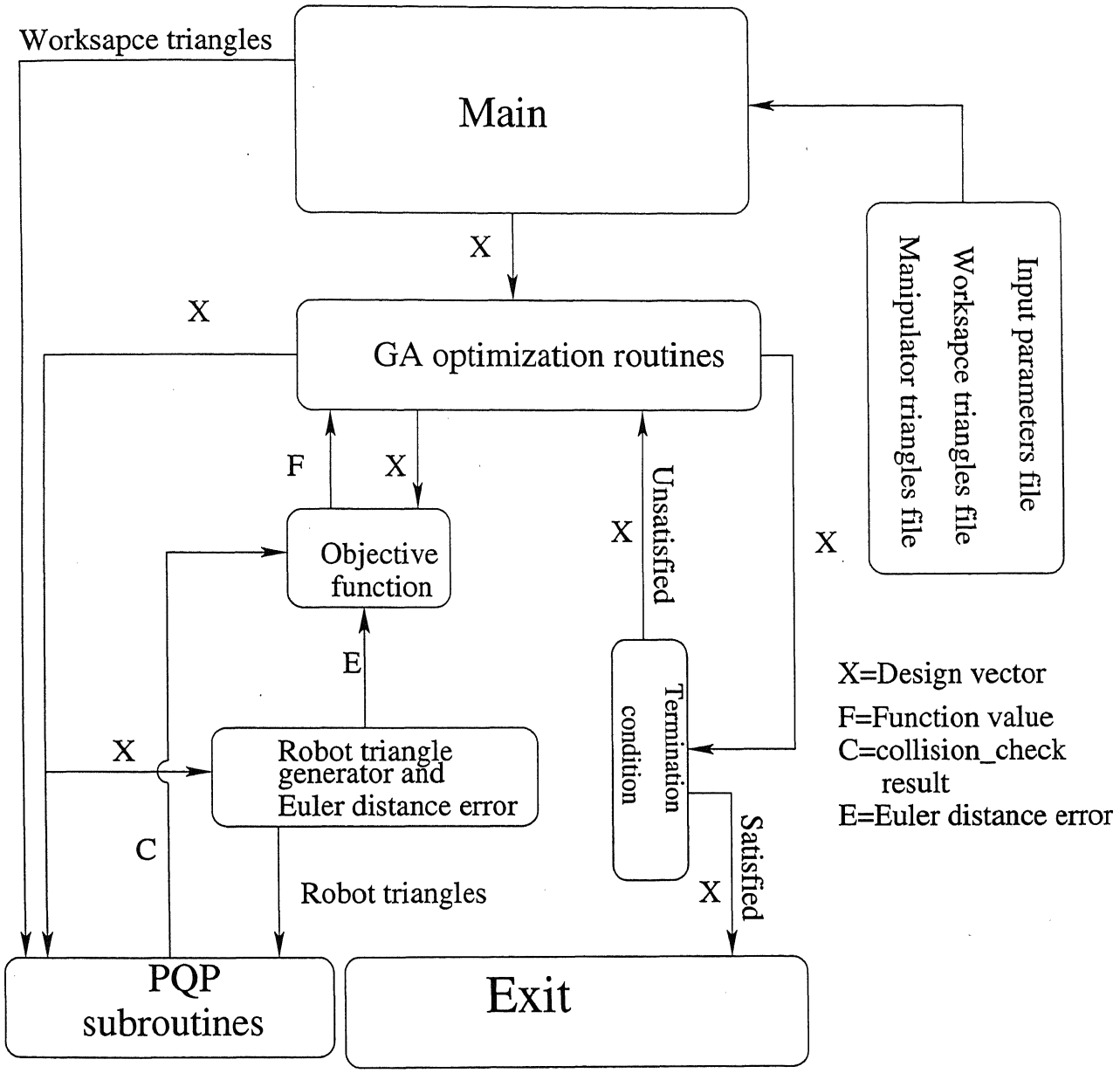


Figure B.1: Flowchart of the Real GA based code

B.3 Template of Input File for Genetic Algorithm Based Code

Half width of the manipulator link(tt) :1/20 to 1/10 of max link length

Coordinate of manipulator base :xb yb zb

No. of links :

No. of prismatic joints :

Position of prismatic joint :

No. of destination points :

Destination points x,y,z... :

How many generations ? ———— :

Population Size ? ———— :

Cross Over Probability ? (0 to 1) :

Mutation Probability ? (0 to 1) — :

Number of variables (Maximum 20) — :

Binary or Real-coded parameters(b or r) :

Lower and Upper bounds of x[1] to x[N] :

Are these bounds rigid ? (y/n) :

String length ? :

Sharing ? (y/n) σ_{share} if yes :

Reports to be printed ? (y/n) :

How many runs ? :

Enter selection opeartor

1. Tournament selection (min or max set by MINM in the code)

2. Roulette wheel selection (always max)

3. Stochastic remainder roulette wheel selection (max)

Your Choice :

Enter tournament size for selection :

Type of cross-over strategy.

1. Crossover in one variable
2. Crossover in all variables
3. Crossover on a straight line

Your choice :

Type of cross over ? (s/b) :

Give eta for Simulated Binary Crossover and Mutation :

Random seed number :

Bibliography

[1] Aho, A.V., Hopcroft, J.E., and Ullman, J.D. *Data Structure and Algorithms*, Addison-Wesely, Reading, MA.(1983).

[2] Sedgewick, P. *Algorithms*, 2^{nd} edition, Addison Wesely. (1988).

[3] Lenarčič, J. "On the execution of the secondary task of redundant manipulators", *ELSEVIER, Robotics and Autonomous Systems*, Vol. 30, 231-236 (2000).

[4] Nakamura, Y., Hanafusa, H. and Yoshikawa , T., "Task-priority based redundancy control of robot manipulators", *Int. J. Robotics Res.*, **6**(2), 3-15 (1987).

[5] Waldron J. Kenneth and Reidy John, "A Study of a Kinematically Redundant Manipulator Structure", *IEEE Proceedings (1986)*

[6] Klein, C.A., "Use of redundancy in the design of robotic systems", *Proc. 2nd Int. Symp. on Robotics Research*, Kyoto, Japan, August 1984.

[7] Yoshikawa, T., "Manipulability of robot mechanisms", *Int. J. Robotics. Res.*, 4(2),pp. 3-9 (1985)

[8] Yoshikawa, T., "Dynamics manipulability of robot manipulators", *J. Robotic. Syst.*, **2**(1), pp. 113-124 (1985).

[9] Maciejewski, C.A. and Klein, C.A., "Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments", *Int. J. Robotics. Res.*, 4(3), pp. 109-117 (1985).

[10] Baillieul, J. "Design of kinematically redundant mechanisms", *Proc. 24th IEEE Conf. on Design and Control*, Ft.Lauderdale, FL, pp. 18-21 (December 1985).

[11] Lee, S. and Lee, J.M., "Reconfiguration of a Redundant Manipulator with Task-Oriented Manipulability", *Int. J. of Robotics and Auto. IEEE*, pp. 1239-1244,(1991).

[12] Paredis, C.J.J. and Khosla, P.K., "Kinematic Design of Serial Link manipulators From Task Specifications", *The Int. J. of Robo. Res.*, Vol. 12, No. 3, June 1993.

[13] Paredis C.J.J. and Khosla, P.K.,"Synthesis Methodology for Task Based Reconfiguration of Modular Manipulator Systems", *Proc. of 6th International Symposium on Robotics Res.*, 2-5 Octobor, 1993.

[14] English, J.D. and Maciejewski, A.A., " Fault tolerance for kinetically redundant manipulators: Anticipating free-swinging joint failures", *IEEE Transaction on Robotics and Automation*, Vol.14 No. 4, August 1998.

[15] Lewis, C.L. and Maciejewski, A.A. " Fault tolerant operation of kinematically redundant manipulators for locked joint failures", *IEEE, Transaction Robotics and Automation* Vol.13, No. 4, (622-629) August 1997.

[16] Singh, J.R. and Rastegar, J., " Optimal Synthesis of Robot manipulators Based on Global Kinematic Parameters", *Mech. Mach. Theory*, Vol. 30, No. 4, pp. 569-580, (1995).

[17] Mariappan, J. and Krishnamurthy, S. "A Generalized Exact Gradient Method for Mechanism Synthesis", *Mech. Mach. Theory*, Vol. 31, No. 4, pp. 413-421, (1996).

[18] Jimenez, J.M., Alvarez, G., Cardenal, J. and Cuadrado, J., "A Simple and General Method for Kinematic Synthesis of Spatial Mehcanisms", *Mech. Mach. Theory*, Vol. 32, No. 3, pp. 323-341, (1997).

[19] Ding Han and Chan Sai Pu," Collision-free Motion Planning for Redundant Robots Using Neural-network Processing", *Engg. Applic. Arti. Intell.*, Vol. 10, No. 2, pp. 179-188, (1997).

[20] Wenger Philippe, "Some guidelines for the kinematic design of new manipulators",*Mech. Mach. Theory* Vol. 35, pp. 437-449, (2000).

[21] Parker, J., Khoogar, A. and Goldberg, D., *Proc. IEEE Int. Conf. on Robotics and Automation,* pp. 679-683 (1986).

[22] Nearchou, A.C. and Aspragathos, A.N.," Application of Genetic Algorithms to Point-to-Point Motion of Redundant manipulators", *Mech. Mach. Theory*, Vol. 31, No. 3, pp. 261-270, (1996).

[23] Larsen, E., Gottschalk, S., Lin, M.C. and Manocha, D.,"Fast Proximity Queries with Swept Sphere Volumes", *Technical Report TR99-018, Department of Computer Science, UNC Chapel Hill.*

[24] Gottschalk, S., Lin, M.C. and Manocha, D.,"OBB-Tree: A Hierarchical Structure for Rapid Interference Detection", *Technical report TR96-013, Department of Computer Science, UNC Chapel Hill.*

[25] Powell, M.J.D.,"A method for nonlinear constraints in minmization problems", In *Optimization*, pp. 283-298. R. Fletcher, ed., Academic Press, (1968).

[26] Hestenes, M.R., "Multiplier and gradient methods", *Journal of Optimization Theory and Application*,4:303-320, (1968).

[27] Rockafellar, R. Tyrrell,"Penalty Methods and Augmented Lagrangians in Nonlinear Programming",*In Fifth Conference on Optimization Techniques, R. Conti and A. Ruberti (eds.)*, Springer-Verlag, 1973, 518-525.

[28] Belegundu, A.D. and Chandraputla, T.R., "Optimization Concept and Applications in Engineering", *Pearson Education(Singapore) Pte. Ltd.*, Indian Branch, (2002).

[29] Holland, J.H. "Adaptation in natural and Artificial Systems", *Ann Arbor*, MIT Press (1975).

[30] Deb, K."When will genetic algorithms work?", *In P. K. Roy and S. D. Mehta (Eds.), Proceedings of the Symposium on Genetic Algorithms, (Dehradun, India)*, pp. 5-22, (1995).

[31] Goldberg, D.E.,"Genetic Algorithms in Search", *Optimization and Machine Learning*. Addison-Wesley (1989).

[32] Deb, K. "Multiobjective Optimization using Evolutionary Algorithms", *John Willey & Sons Ltd.*, April 2002.

[33] Goldberg, D.E. and Deb, K."A comparison of selection schemes used in genetic algorithms", *Foundations of Genetic Algorithms*, I, 69-93 (1991).

[34] Lin, M.C. and Canny, John F., "Efficient algorithms for incremental distance computation",*In IEEE Conference on Robotics and Automation*, pp. 1008-1014, (1991).

[35] Craig, J.J., "Introduction to Robotics Mechanics and Control", *Addision- Wesley Longman Inc.*, Second Edition.